Doctoral Thesis

Consensus Maximization: Theoretical Analysis and New Algorithms

Submitted by:

Zhipeng Cai

Supervised by:

Assoc. Prof. Tat-Jun CHIN

Prof. David SUTER

A thesis submitted in fulfillment of the requirements for the degree of Doctor of Philosophy

in the

Faculty of Engineering, Computer and Mathematical Sciences

School of Computer Science

September 2020

Declaration

I certify that this work contains no material which has been accepted for the award of any other degree or diploma in my name, in any university or other tertiary institution and, to the best of my knowledge and belief, contains no material previously published or written by another person, except where due reference has been made in the text. In addition, I certify that no part of this work will, in the future, be used in a submission in my name, for any other degree or diploma in any university or other tertiary institution without the prior approval of the University of Adelaide and where applicable, any partner institution responsible for the joint-award of this degree.

I acknowledge that copyright of published works contained within this thesis resides with the copyright holder(s) of those works.

I also give permission for the digital version of my thesis to be made available on the web, via the University's digital research repository, the Library Search and also through web search engines, unless permission has been granted by the University to restrict access for a period of time.

I acknowledge the support I have received for my research through the provision of a scholarship from the China Scholarship Council.

Signature:

Date: 11-Feb-2020

THE UNIVERSITY OF ADELAIDE

Abstract

School of Computer Science

Doctor of Philosophy

Consensus Maximization: Theoretical Analysis and New Algorithms

by Zhipeng Cai

The core of many computer vision systems is model fitting, which estimates a particular mathematical model given a set of input data. Due to the imperfection of the sensors, pre-processing steps and/or model assumptions, computer vision data usually contains *outliers*, which are abnormally distributed data points that can heavily reduce the accuracy of conventional model fitting methods. Robust fitting aims to make model fitting insensitive to outliers. *Consensus maximization* is one of the most popular paradigms for robust fitting, which is the main research subject of this thesis.

Mathematically, consensus maximization is an optimization problem. To understand the theoretical hardness of this problem, a thorough analysis about its computational complexity is first conducted. Motivated by the theoretical analysis, novel techniques that improve different types of algorithms are then introduced.

On one hand, an efficient and deterministic optimization approach is proposed. Unlike previous deterministic approaches, the proposed one does not rely on the relaxation of the original optimization problem. This property makes it much more effective at refining an initial solution.

On the other hand, several techniques are proposed to significantly accelerate consensus maximization tree search. Tree search is one of the most efficient global optimization approaches for consensus maximization. Hence, the proposed techniques greatly improve the practicality of globally optimal consensus maximization algorithms.

Finally, a consensus-maximization-based method is proposed to register terrestrial LiDAR point clouds. It demonstrates how to surpass the general theoretical hardness by using special problem structure (the rotation axis returned by the sensors), which simplify the problem and lead to application-oriented algorithms that are both efficient and globally optimal.

Acknowledgements

This page represents my sincere appreciation to those important people during my Ph.D career.

Of course, the first one to thank is my supervisor, Assoc. Prof. Tat-Jun Chin. T.J. (that is how we call him) was like a sculpturer, who carefully shaped me from a student to a researcher. Discussing ideas with him is certainly one of my most enjoyable things.

I would like to also thank my co-supervisor, Prof. David Suter, for the discussions and paper revisions. His passion about research let me understand how exciting science can really be.

I would like to thank Dr. Vladlen Koltun, who, as the mentor of my internship, and also a friend, has provided a lot of useful advices to my life and research.

I would like to thank Prof. Konrad Schindler, who provided critical suggetions to our work and carefully modified the paper during our collaboration.

I would like to thank Dr. Alvaro Parra Bustos and Dr. Huu Le, who passed me a lot of experience and kindly helped me in multiple reserach projects.

I would like to thank Dr. Nan Li and Dr. Pulak Purkait, with whom I had inspiring discussions about consensus maximization tree search.

I would like to thank Gang Huang and Yineng Wang, who have provided me a familiy-like accommondation and helped me a lot during these years.

I would like to thank my friends (names are saved in my heart since there are too many to be fit into a single page), with whom I have created so many unforgettable memories.

Finally, I would like to express my most sincere thanks to my parents for their unconditional and consistent support, without which I can never have such a wonderful research career.

Contents

Publications

This thesis is in part result of the work presented in the following papers:

- Tat-Jun Chin, Zhipeng Cai, Frank Neumann: Robust fitting in computer vision: easy or hard? European Conference on Computer Vision (ECCV) 2018.
- Zhipeng Cai, Tat-Jun Chin, Huu Le, David Suter: Deterministic consensus maximization with biconvex programming. European Conference on Computer Vision (ECCV) 2018.
- Zhipeng Cai, Tat-Jun Chin, Alvaro Parra Bustos, Konrad Schindler: Practical Optimal Registration of Terrestrial LiDAR Scan Pairs. ISPRS Journal of Photogrammetry and Remote Sensing 2019.
- Zhipeng Cai, Tat-Jun Chin, Vladlen Koltun, Consensus Maximization Tree Search Revisited, International Conference on Computer Vision (ICCV) 2019.
- Tat-Jun Chin, Zhipeng Cai, Frank Neumann: Robust fitting in computer vision: easy or hard? International Journal on Computer Vision (IJCV) 2019.

Chapter 1

Introduction

1.1 Robust fitting in computer vision

Computer vision, as its name implies, aims to mimic the human visual system on computers. An important tool to achieve this goal is *model fitting*, which is used to estimate some model given a set of input data. The model refers to some mathematical quantities that we are interested in. For example, the pose of an object, the trajectory of a moving target, the 3D structure of a scene and so on. In computer vision, the input for estimating the model is often a set of visual data, such as images, videos, 3D models and so on.

Robust fitting is a special model fitting technique designed to handle the case where input data is contaminated by *outliers*. In statistics, outliers refer to the data points that differs significantly from others [1]. The existence of outliers can make non-robust model fitting methods highly biased. To demonstrate this effect, we show in Figure 1.1 a line fitting problem with one outlier.

In this problem, we want to fit a line on a set of 2D points $\mathcal{S} = \{\mathbf{s}_i\}_{i=1}^{n}$, where $\mathbf{s}_i = \begin{bmatrix} s_i^{(1)} \\ s_i^{(2)} \end{bmatrix}$. Given a candidate solution $\boldsymbol{\theta} = \begin{bmatrix} \theta^{(1)} \\ \theta^{(2)} \end{bmatrix}$, its relationship with a point $\mathbf{s}_i$ can be represented by:

$$s_i^{(2)} = s_i^{(1)} \cdot \theta^{(1)} + \theta^{(2)} + \eta_i, \tag{1.1}$$

where η_i is the noise introduced by the data generation process.

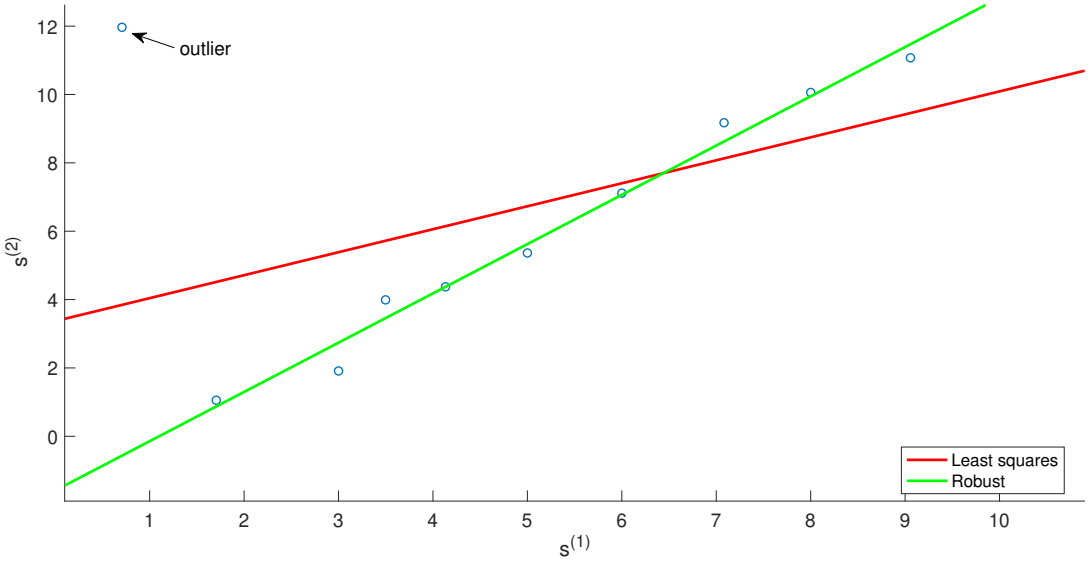


FIGURE 1.1: Line fitting example. The outlier (the upper left point) does not follow the linear distribution of other points. The Least Squares solution is far away from the desired location due to the existence of outliers.

The simplest model fitting strategy is Least Squares [2], which estimates the model by minimizing the sum of squared residuals, i.e.,

$$f_{\mathrm{LS}}(\boldsymbol{\theta}|\mathcal{S}) = \sum_{\mathbf{s}_i \in \mathcal{S}} r(\boldsymbol{\theta}|\mathbf{s}_i)^2, \tag{1.2}$$

where each residual $r(\boldsymbol{\theta}|\mathbf{s}_i)$ is defined as the magnitude of the noise η_i, i.e.,

$$r(\boldsymbol{\theta}|\mathbf{s}_i) = |s_i^{(2)} - (s_i^{(1)} \cdot \theta^{(1)} + \theta^{(2)})|. \tag{1.3}$$

As a non-robust model fitting methods, Least Squares is highly sensitive to outliers. From the Baysian perspective, minimizing $f_{\mathrm{LS}}(\boldsymbol{\theta}|\mathcal{S})$ is essentially performing maximum likelihood estimation [3], with the assumption that the noise magnitude η_i follows a normal distribution, and is independent and identically distributed (i.i.d.). However, this assumption is violated due to the extreme distribution of outliers. Because the noise level η_i of the outlier is high w.r.t. the desired line (rendered in green), the Least Squares solution is biased towards the outlier to minimize (1.2). This problem motivates the need of robust fitting techniques, which are designed to be insensitive to outliers.

Many computer vision applications rely on robust fitting. These include estimating geometric primatives (e.g., lines, planes, ellipses), and more complex tasks such as object recognition, Structure-from-Motion and navigation.

Outliers exist in these applications because it is hard to perfectly model every detail of the data generation process. For example, the data acquisition sensors are not always

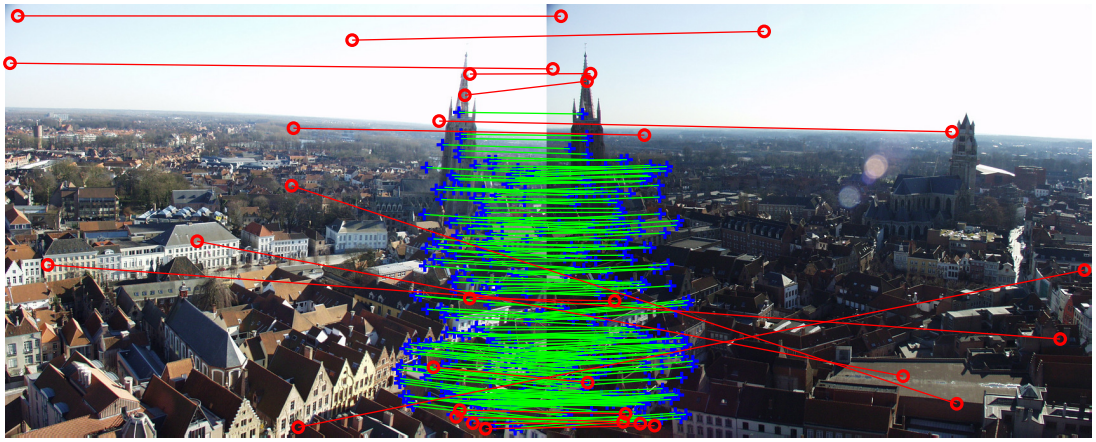


FIGURE 1.2: Example data for two-view geometry. To estimate the transformation between two images, feature matching algorithms are executed first to extract candidate correspondences. These candidates are rendered as lines between the two images. Note that the red lines are wrong matches, which are the outliers.

reliable, they may sometimes return erroneuous measurments. Meanwhile, most sensors cannot directly and only capture the data that we need for model fitting. Often a preprocessing step is required to filter out irrelavant measurements and extract quantities of interests. Such step often introduces outliers.

Figure 1.2 shows the data for two-view geometry. In this example, the task is to estimate the transformation that aligns corresponding pixels (that capture the same scene points) of the two images. As a preprocessing step, some feature matching algorithm [27, 14] is first executed to extract candidate correspondences. The outliers in this case are those incorrect candidates (red lines).

Point set registration is another example, where we want to estimate the relative rotation and translation between two sets of points. Figure 1.3 shows example data in 3D. Similar as the case of two-view geometry, 3D feature matching algorithms [36, 35] are required to extract candidate correspondences, which introduce outliers. Note that in this case, due to the instability of 3D feature matching algorithms, most of the candidates are outliers.

1.2 Robust criteria

One important question in robust fitting algorithms is the design of robust criteria, which is a cost/loss function that is insensitive to outliers and is used to replace $f_{\text{LS}}(\boldsymbol{\theta}|\mathcal{S})$ in (1.2).

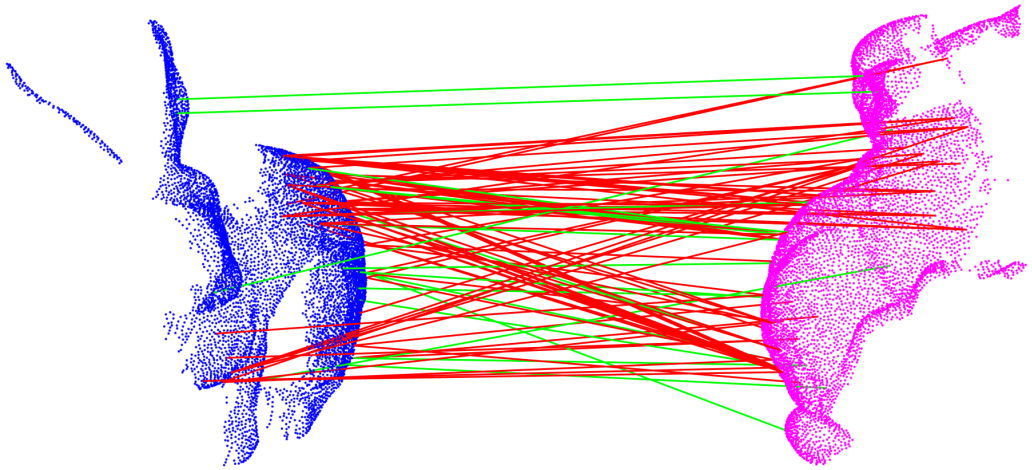


FIGURE 1.3: Example data for 3D point set registration. The lines between points are candidate correspondences found by 3D feature matching algorithms. The red lines are the outliers.

1.2.1 Consensus maximization

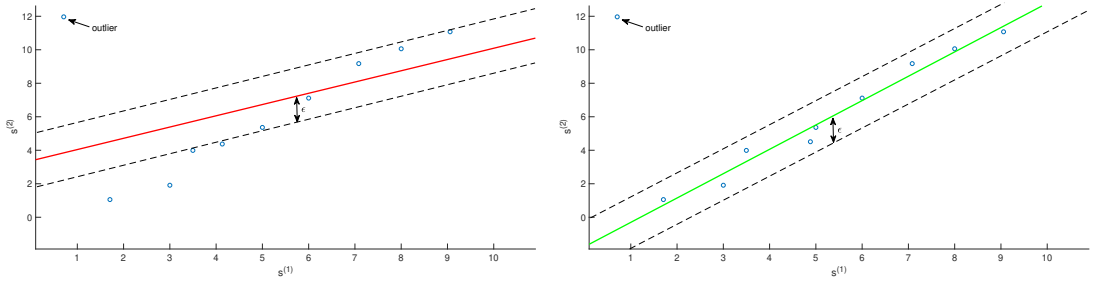
In computer vision, one of the most widely used robust fitting criteria is *consensus maximization*, where we want to maximize the *consensus*, which is defined as

$$f_{\text{MC}}(\boldsymbol{\theta}|\mathcal{S}) = \sum_{\mathbf{s}_i \in \mathcal{S}} \mathbb{I}\{r(\boldsymbol{\theta}|\mathbf{s}_i) \leq \epsilon\}. \tag{1.4}$$

In (1.4), $\mathbb{I}\{\cdot\}$ is an indicator function that returns 1 if the condition inside is true, and 0 otherwise; ϵ is a predefined value called the *inlier threshold*, which estimates the maximum residual of an inlier (the data point that is not an outlier) w.r.t. the desired model.

Intuitively, $f_{\text{MC}}(\boldsymbol{\theta}|\mathcal{S})$ counts the number of data points that are consistent with the model $\boldsymbol{\theta}$. To show why this is a reasonable objective to maximize, we re-use the line fitting example in Figure 1.1. Figure 1.4 shows the consensus of the Least Squares solution and the green line in Figure 1.1. We can see that with a reasonable inlier threshold ϵ, the green line, which is the desired one, has the maximum consensus value.

One may argue that there can be multiple lines around the green one (e.g., the lines slightly above and below it) that have the same consensus. Indeed, this is often the case. In computer vision, the purpose of consensus maximization lies more in the identification of inliers or the removal of outliers. Given a solution $\boldsymbol{\theta}^*$ that maximizes $f_{\text{MC}}(\boldsymbol{\theta}|\mathcal{S})$, we label all points that are consistent with $\boldsymbol{\theta}^*$ as inliers. To refine the accuracy of the fitted model, Least Squares on the indentified inliers is often conducted after consensus maximization.



(a) Least Squares solution (Consensus = 6). (b) Desired solution (Consensus = 9).

FIGURE 1.4: Consensus of the Least Squares solution and the desired one. Points between the dashed line have residuals that are smaller than ϵ.

The main focus of this thesis is the theories and applications of consensus maximization. But for completeness, we also briefly introduce in the following sections, other robsut criteria and their relationship with consensus maximization.

1.2.2 M-estimator

As mentioned in Chapter 1.1, Least Squares is non-robust because of its i.i.d. and normal distribution assumption. Motivated by this problem, many robust criteria that can tolerate outlying distributions are proposed in the field of statistics.

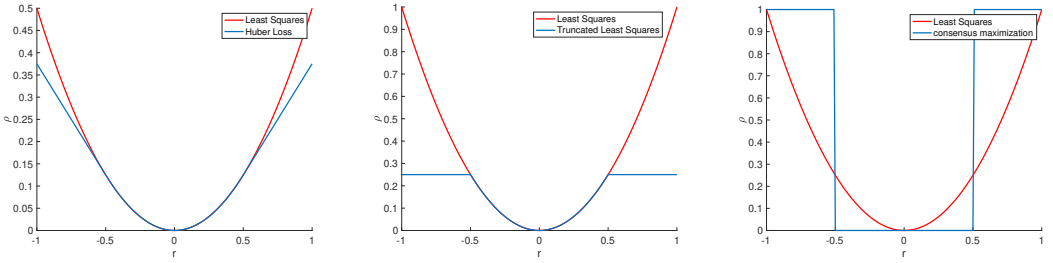
M-estimators [4] are a class of criteria that generalize the maximum likelihood estimator. They minimizes the following cost function:

$$f_{\text{ME}}(\boldsymbol{\theta}|\mathcal{S}) = \sum_{\mathbf{s}_i \in \mathcal{S}} \rho(r(\boldsymbol{\theta}|\mathbf{s}_i)), \tag{1.5}$$

where $\rho(\cdot)$ is a function of $r(\boldsymbol{\theta}|\mathbf{s}_i)$ that is symmetric, positive definite and has a unique minimum of 0 at $r(\boldsymbol{\theta}|\mathbf{s}_i) = 0$. Intuitively, $\rho(\cdot)$ controlls the influence of each residual to the model estimation. Under this definition, we can see that Least Squares is also an m-estimator, though non-robust. The $\rho(\cdot)$ in robust m-estimators is designed to reduce the effect of large residuals, and in turn the outliers.

For example, in Huber Loss [18] (see Figure 1.5(a)),

$$\rho(r(\boldsymbol{\theta}|\mathbf{s}_i)) = \begin{cases} \frac{1}{2} r(\boldsymbol{\theta}|\mathbf{s}_i)^2, & r(\boldsymbol{\theta}|\mathbf{s}_i) \leq \epsilon \\ \epsilon(r(\boldsymbol{\theta}|\mathbf{s}_i) - \frac{1}{2}\epsilon), & \text{otherwise} \end{cases} \tag{1.6}$$



(a) Huber Loss. (b) Truncated Least Squares. (c) Consensus Maximization.

FIGURE 1.5: ρ functions in M-estimators ((a) and (b)) and consensus maximization ((c)). The horizontal axis is the value of $r(\boldsymbol{\theta}|\mathbf{s}_i)$. We use $\epsilon = 0.5$ in this example. The (re-scaled) cost function of Least Squares is also plotted for better comparison.

The growth rate of $\rho(r(\boldsymbol{\theta}|\mathbf{s}_i))$ reduces from quadratic to linear when $r(\boldsymbol{\theta}|\mathbf{s}_i)$ is larger than the inlier threshold ϵ. Though more robust than the squared residual, Huber Loss is still unbounded, meaning that extreme outliers can still have catastrophic effect.

Redescending m-estimators [5] are a sub-class of m-estimators with higher robustness. Their main difference to Huber Loss is that instead of linearly growing, the function values remain constant for large residuals. For example, the Truncated Least Squares [17] (see Figure 1.5(b)) can be expressed as:

$$\rho(r(\boldsymbol{\theta}|\mathbf{s}_i)) = \begin{cases} r(\boldsymbol{\theta}|\mathbf{s}_i)^2, & r(\boldsymbol{\theta}|\mathbf{s}_i) \leq \epsilon \\ \epsilon^2. & \text{otherwise} \end{cases} \tag{1.7}$$

Due to the bounded effect of outliers, redescending m-estimators can tolerate a large amount (close to 50%) of outliers.

As shown in Figure 1.5(c), though consensus maximization can be viewed as setting $\rho(r(\boldsymbol{\theta}|\mathbf{s}_i)) = 1 - \mathbb{I}\{r(\boldsymbol{\theta}|\mathbf{s}_i) \leq \epsilon\}$, it is not an m-estimator, since its minimum is not unique.

1.2.3 Least Median Squares and Least k-th Order Statistics

Besides m-estimators, Least Median Squares (LMS) [33] is also widely used in robust statistics. As the name suggests, it minimizes the median of the residuals (squared or not squared are equivalent), i.e.,

$$\underset{\mathbf{s}_i \in \mathcal{S}}{\text{median}} \; r(\boldsymbol{\theta}|\mathbf{s}_i). \tag{1.8}$$

LMS is tolerable to 50% outliers.

A natural generalization of LMS is the Least k-th Order Statistics (LkOS) estimator, where we want to minimize the k-th smallest residual. Interestingly, there is a "complexity-equivalence" between LkOS and consensus maximization. Specifically, if we can solve one of these two problems in polynomial time, the other one can also be solved in polynomial time. And this "equivalence" leads to the NP-hardness proof of consensus maximization, see Chapter 3 for more details.

1.3 Example residual forms

Depending on the application, the residual $r(\boldsymbol{\theta}|\mathbf{s}_i)$ in robust criteria can have different forms.

For example, in the case of homography estimation (see Figure 1.2), the input data $\mathcal{S}$ is a set of feature matches, where each $\mathbf{s}_i = \{\tilde{\mathbf{p}}_i, \tilde{\mathbf{q}}_i\}$ consists of the homogeneous coordinates of the two matched pixels $\mathbf{p}_i \in \mathbb{R}^2$ and $\mathbf{q}_i \in \mathbb{R}^2$. The homogeneous coordinate of a pixel $\mathbf{x} \in \mathbb{R}^2$ is defined as $\begin{bmatrix}\mathbf{x} \\ 1\end{bmatrix}$. The model we want to estimate in this case is a 3×3 homography matrix $\mathbf{H}(\boldsymbol{\theta})$, where we put $\boldsymbol{\theta}$ in the bracket to indicate that $\mathbf{H}$ is parametrized by the vector $\boldsymbol{\theta} \in \mathbb{R}^8$. The residual in this case is

$$r(\boldsymbol{\theta}|\mathbf{s}_i) = \left\| \frac{\mathbf{H}(\boldsymbol{\theta})^{(1:2)}\mathbf{p}_i}{\mathbf{H}(\boldsymbol{\theta})^{(3)}\mathbf{p}_i} - \mathbf{q}_i \right\|_2, \tag{1.9}$$

where $\mathbf{H}(\boldsymbol{\theta})^{(1:2)}$ is the first two rows of $\mathbf{H}(\boldsymbol{\theta})$ and $\mathbf{H}(\boldsymbol{\theta})^{(3)}$ is the last row.

In the case of point cloud registration (see Figure 1.3), $\mathbf{s}_i = \{\mathbf{p}'_i, \mathbf{q}'_i\}$ is the coordinates of the matched 3D points. And the model we want to estimate is the rigid transformation (rotation $\mathbf{R}(\boldsymbol{\theta})$ and translation $\mathbf{t}(\boldsymbol{\theta})$ parametrized by $\boldsymbol{\theta} \in \mathbb{R}^6$) between two point clouds. The residual in this case is

$$r(\boldsymbol{\theta}|\mathbf{s}_i) = \left\| \mathbf{R}(\boldsymbol{\theta})\mathbf{p}'_i + \mathbf{t}(\boldsymbol{\theta}) - \mathbf{q}'_i \right\|_2, \tag{1.10}$$

1.4 Optimization algorithms for robust fitting

After choosing a robust criterion, the last question in the design of robust fitting algorithms is how to find the best model w.r.t. this criterion. Unlike Least Squares, which can be efficiently solved by gradient-based methods or even in closed-form, optimization for robust criteria is often much harder (See Chapter 3 for more details).

This section briefly introduces optimization algorithms for the robust criteria mentioned in Chapter 1.2. A detailed survey focusing on consensus maximization algorithms can be found in Chapter 2.

1.4.1 Random-sampling-based algorithms

Random-sampling-based algorithms are widely used in practice, and with a long history.

RANSAC [16], proposed by Fischler et al. in 1981, is an important representative of this class. It is the first algorithm for consensus maximization. RANSAC iteratively fits models on randomly sampled data subsets and returns the model with the highest consensus. The key insight is that the iteration number quantifies the confidence of sampling a *full-inlier* subset (see Chapter 2.1.1 for more details). And the basic assumption of RANSAC is that, a model fit on a full-inlier subset tends to have a high consensus. Many RANSAC variants [12, 38, 11, 23, 39] were proposed later on to further improve the performance.

Similar to RANSAC, an algorithm called "program for robust regression (PROGRESS)" [34, Chapter 5] was also proposed for LMS. The key of this algorithm is also a random sample-and-test process.

Random-sampling-based algorithms are widely used because of their practical efficiency. However, the inherent randomness makes them hard to guarantee the solution quality and sometimes unstable.

1.4.2 Gradient-based deterministic optimization algorithms

The drawback of random-based agorithms motivates the need of gradient-based deterministic optimization algorithms, which update the model in a more guided fashion.

The standard optimization algorithm for m-estimators is *Iterative Re-weighted Least Squares* (IRLS), which was first proposed in 1970's [40]. Given an initial solution $\boldsymbol{\theta}_0$, IRLS alternates between *weight assignment* and *weighted Least Squares* until convergence (see [7] for the proof of convergence). The weight assignment step attaches to each data point a weight defined as

$$w(\boldsymbol{\theta}_0|\mathbf{s}_i) = \frac{\partial \rho(r(\boldsymbol{\theta}_0|\mathbf{s}_i))}{\partial r(\boldsymbol{\theta}_0|\mathbf{s}_i)} \cdot \frac{1}{r(\boldsymbol{\theta}_0|\mathbf{s}_i)}. \tag{1.11}$$

And the weighted Least Squares step updates $\boldsymbol{\theta}_0$ by:

$$\boldsymbol{\theta}_0 \leftarrow \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \sum_{\mathbf{s}_i \in \mathcal{S}} w(\boldsymbol{\theta}_0|\mathbf{s}_i) \cdot r(\boldsymbol{\theta}|\mathbf{s}_i)^2. \tag{1.12}$$

Though IRLS is widely used for m-estimators, it cannot be directly applied to consensus maximization. This is because the objective for consensus maximization is a step function (see Figure 1.4), hence the gradient $\frac{\partial \rho(r(\boldsymbol{\theta}|\mathbf{s}_i))}{\partial r(\boldsymbol{\theta}|\mathbf{s}_i)} = 0$ for almost all $\boldsymbol{\theta}$. This makes the update step (1.11) always return 0 weight for all $\mathbf{s}_i$.

The first gradient-based consensus maximization algorithm was proposed by Le et al. [20] at 2017. They first relaxed the objective $f_{\text{MC}}(\boldsymbol{\theta}|\mathcal{S})$ of consensus maximization using ℓ_1 penalization, and then performed optimization on the relaxed objective function. Several approaches [22, 21, 32] that utlize different relaxation techniques were proposed later. Compared to RANSAC variants, gradient-based algorithms conduct a more guided search. Hence, they are often more effective in refining an initial solution. However, the use of relaxation makes these algorithms prone to the tuning of smoothing parameters, which is time consuming. Moreover, incorrect settings of smoothing parameters can even make these algorithms return solutions that are worse than the initial one (see Chapter 4 for more details).

1.4.3 Globally optimal algorithms

Though efficient, algorithms in Chapter 1.4.1 and 1.4.2 are all sub-optimal, i.e., they cannot guarantee to find the best solution possible. This motivates the need of globally optimal algorithms.

Branch-and-Bound is a standard global optimization technique. It was proposed by Land and Doig [19] in 1960 and later widely used for robust fitting [26, 41, 31]. The key idea is divide-and-conquer. Specifically, the parameter space of the model $\boldsymbol{\theta}$ is iteratively divided, with upper and lower bounds of the optimal objective value computed over divided sub-spaces. With the increase of the division resolution, the upper and lower bounds will eventually meet, and the solution corresponding to the final bounds is the optimal one. With properly designed bounding functions, Branch-and-Bound can be used for any robust criterion mentioned in previous chapters. However, the bounding functions are heavily application-dependent and not always trivial to construct.

Recently, the development of globally optimal consensus maximization algorithms has received a lot of attention. Utilizing special properties of consensus maximization, algorithms such as basis enumeration [29, 15] and tree search [25, 10] (see Chapter 2 for

more details) were proposed. In practice, tree search has been demonstrated to be one of the most efficient globally optimal algorithms.

1.5 Open questions and contributions

1.5.1 How hard is robust fitting?

Recent advancement in globally optimal consensus maximization [26, 41, 31, 29, 15, 25, 10] seems to provide a positive signal on the tractability of globally optimal robust fitting. Empirical efficiency has been demonstrated in several existing works, which raises questions that motivate the work of this thesis.

Specifically, *how hard is consensus maximization? Can we expect an efficient and globally optimal algorithm in general?* To answer these questions, we first study in Chapter 3 the theoretical hardness of consensus maximization. The results are briefly summarized as follows:

1. Consensus maximization is NP-hard. This means that, there are no algorithms that can solve consensus maximization in time $\text{poly}(|\mathcal{S}|, d)$, where $|\mathcal{S}|$ is the number of input data points, and d is the dimension of $\boldsymbol{\theta}$, and $\text{poly}(\cdot)$ is a polynomial function.
2. Consensus maximization is W[1]-hard in the dimension d. This means that, there are no algorithms that can solve consensus maximization in time $f(d)\text{poly}(|\mathcal{S}|)$, where $f(d)$ is an arbitrary function of d.
3. Consensus maximization is APX-hard. This means that, there are no polynomial time algorithms that can find an approximate solution with consensus up to $(1-\delta)\Psi^*$ for any known *approximation ratio* δ, where Ψ^* is the maximum consensus.

The NP-hardness negates the existance of a general and tractable globally optimal algorithm. The W[1]-hardness further tells us that *existing globally optimal algorithms must all scale exponentially w.r.t.* d, and hence are unlikely to remain efficient when d grows.

The APX-hardness further prohibits the development of efficient approximate algorithms. Note that none of the approaches mentioned in Chapter 1.4.1 and 1.4.2 can guarantee an arbitrary approximation ratio δ, i.e., they are not approximate algorithms in this important technical sense.

1.5.2 What kind of algorithms to develop?

The hardness results also provide indications to the algorithms we should develop in the future.

The APX-hardness indicates that the development of efficient and deterministically convergent algorithm is a good direction to pursue, since it is unlikely to obtain guarantees to the solution quality. Chapter 4 proposes a more effective deterministic algorithm. Unlike previous algorithms of the same type [20, 22, 21, 32], the proposed one does not relax the original objective function, and hence is free from the tuning of smoothing parameters.

A slightly positive result also proved in Chapter 5 is that:

- Consensus maximization is Fixed-Parameter Tractable (FPT) w.r.t. the number of outliers o. Specifically, it can be solved in time $\mathcal{O}((o+1)^{d+1}\text{poly}(|\mathcal{S}|, d))$

Intuitively, this means that when o is small, consensus maximization can still be efficient solved globally, even though $|\mathcal{S}|$ is large. This result is proved by constructing a tree search algorithm that is FPT. This suggests that among all globally optimal algorithms, accelerating tree search is a more promising direction. Chapter 5 proposes new techniques that further accelerate tree search, making it capable of handling a much larger o.

Meanwhile, although a general algorithm does not exist, we can still utilize special problem structures to design application-oriented approaches that are both efficient and globally optimal. Chapter 6 proposes such an algorithm for registering LiDAR point clouds, which utilizes the rotation axis obtained from the sensor to simplify the problem and significantly speed up global optimization.

To further improve the practicality of globally optimal algorithms, we can also develop preprocessing algorithms [31, 8], where efficient heuristics are designed to detect and remove some true outliers *before performing global optimization*, so that to reduce the input size for globally optimal algorithms. Though without guarantees: in practice, these preprocessing algorithms can often remove a considerable amount of outiers, making the acceleration to global optimization much more significant than the overhead of preprocessing. Chapter 6 also applies preprocessing to accelerate Branch-and-Bound.

1.6 Thesis outline

The upcoming chapters are organized as follows:

- Chapter 2 provides a detailed discussion about existing works for consensus maximization.
- Chapter 3 discusses the theoretical hardness of consensus maximization. The results of the chapter distinguish the performance of existing algorithms, and reveals the possibilities or impossibilities to develop new ones with certain properties.
- Chapter 4 contributes to the improvement of efficient but sub-optimal methods. It proposes a novel deterministic optimization algorithm for consensus maximization. Unlike existing approaches of the same type, the proposed one does not rely on the relaxation of the cost function (1.4), thus is more effective at refining an initial solution.
- Chapter 5 proposes new techiques that can significantly speed up consensus maximization tree search, making globally optimal solutions much more tractable in practice.
- Chapter 6 applies consensus maximization to efficiently and optimally register terrestrial LiDAR point clouds. Though the result of Chapter 3 precludes the development of tractable and globally optimal algorithms in general. We show that this goal is still achievable for some applications, where specific structures can be utilized to simplify the optimization problem.

Chapter 2

Literature Review

Existing algorithms for consensus maximization can be categorized as:

1. Efficient but sub-optimal algorithms.
2. Globally optimal algorithms.

This chapter surveys several representatives of each categories, and discusses their relations to the work of this thesis.

2.1 Efficient but sub-optimal algorithms

2.1.1 RANSAC and its variants

Arguably, RANSAC [16] is the most widely used algorithm for consensus maximization. As its full name (RAndom SAmple Consensus) suggests, RANSAC works by iteratively performing *random-sampling-and-consensus-verification*. In each iteration i, RANSAC fits the model $\boldsymbol{\theta}_i$ on a randomly sampled *minimal* data subset. The word "minimal" refers to the smallest number of data points that can be used to decide a model. E.g., for 2D line fitting (Figure 1.1), we need 2 points to decide a line, then the minimum data subset is two points. After computing each $\boldsymbol{\theta}_i$, RANSAC calculates its consensus $f_{\text{MC}}(\boldsymbol{\theta}_i|\mathcal{S})$ (defined in (1.4)) and replaces the current best model $\boldsymbol{\theta}^*$ with $\boldsymbol{\theta}_i$ if $f_{\text{MC}}(\boldsymbol{\theta}_i|\mathcal{S}) > f_{\text{MC}}(\boldsymbol{\theta}^*|\mathcal{S})$. $\boldsymbol{\theta}^*$ is returned as the final solution when RANSAC terminates.

The basic assumption of RANSAC is that, models fitted on *full-inlier data subsets* (i.e., a subset that contains only inliers) tend to have high consensus. Therefore, RANSAC terminates when we have a high confidence p (often set to be close to 1, e.g., 0.99) that

we have sampled at least one full-inlier minimal subset. And this confidence can be guaranteed by the following termination criterion:

$$i > \frac{\log(1-p)}{\log\left(1-\left(\frac{f_{\mathrm{MC}}(\boldsymbol{\theta}^*|\mathcal{S})}{|\mathcal{S}|}\right)^w\right)}, \tag{2.1}$$

where i is the iteration number, w is the size of the minimal subset, and $p \in (0,1)$ is the required confidence, which is set by the user.

To show why (2.1) guarantees the confidence p of sampling a full-inlier minimal subset, note that when (2.1) is satisfied, we have

$$\log(1-p) > \log\left(\left(1-\left(\frac{f_{\mathrm{MC}}(\boldsymbol{\theta}^*|\mathcal{S})}{|\mathcal{S}|}\right)^w\right)^i\right) \tag{2.2}$$

$$\Rightarrow 1-\left(1-\left(\frac{f_{\mathrm{MC}}(\boldsymbol{\theta}^*|\mathcal{S})}{|\mathcal{S}|}\right)^w\right)^i > p. \tag{2.3}$$

Denote η as the inlier rate of $\mathcal{S}$. Since $\frac{f_{\mathrm{MC}}(\boldsymbol{\theta}^*|\mathcal{S})}{|\mathcal{S}|} \leq \eta$, we have

$$1-(1-\eta^w)^i \geq 1-\left(1-\left(\frac{f_{\mathrm{MC}}(\boldsymbol{\theta}^*|\mathcal{S})}{|\mathcal{S}|}\right)^w\right)^i. \tag{2.4}$$

Combining (2.3) and (2.4), we have when (2.1) is true,

$$1-(1-\eta^w)^i > p. \tag{2.5}$$

Note that $1-(1-\eta^w)^i$ is the possibility that we have sampled at least one full-inlier minimal subset in i iterations.

RANSAC is popular due to its practical efficiency, especially when w is small and/or η is high. This efficiency can be further improved for specific applications [11, 38], where prior knowledge (provided by similarity measurements between feature matches) can be utilized to guide the sampling procedure, such that we can find full-inlier subsets more efficiently.

The drawback of RANSAC lies in the uncertain optimality. Due to the inherent randomness, the solution quality of RANSAC varies in different runs. More importantly, the consensus of RANSAC solutions is often far from the maximum achievable. One reason is that models fitted on minimal samples are very sensitive to the data noise, even though the minimal sample contains only inliers (see [39] for an example).

To alleviate the problem introduced by minimal samples, several RANSAC variants [12, 23, 39] are developed. LO-RANSAC [12, 23] proposes to fit models on *non-minimal* samples once a better solution is discovered. Tran et al. [39] suggests to sample full-inlier

subsets with large spans. Though these algorithms can often improve the solution of RANSAC, they are still based on heuristics and therefore cannot guarantee the improvement, especially on challenging data (see Chapter 4).

2.1.2 Deterministic local optimization algorithms

Motivated by the need of a more stable refinement strategy than heuristics, recent algorithms [20, 21, 32] start to apply deterministic optimization techniques for consensus maximization. These algorithms start from an initial solution (can be obtained for example, by random guess, least squares, or even from the solution of RANSAC), and iteratively perform deterministic updates to improve the solution quality. Compared to random heuristics such as LO-RANSAC, these algorithms are more effective in refining an initial solution, though they still cannot guarantee the global optimality.

The major obstacles for applying deterministic optimization to consensus maximization is that, the objective (1.4) we want to maximize is a step function, which is not smooth, and has zero gradient almost everywhere (see Figure 1 of Chapter 4 for demonstration). Hence, we cannot directly perform gradient-based optimization to improve the initial solution. To overcome this problem, previous algorithms first relax the original objective function (e.g., using ℓ_1 penalization [20], reformulation based on Alternating Direction Method of Multiplier (ADMM) [21], and smooth surrogate functions [32]), and then conduct optimization over the relaxed objective functions. Though these methods enable gradient-based optimization techniques, the use of relaxation inevitably makes them susceptible to the vagaries of smoothing parameters, which controls the degree of relaxation during optimization. The optimal choice of these smoothing parameters often varies between different problems. And as will be demonstrated in Chapter 4, incorrect settings of smoothing parameters can make these methods return a solution that is even worse than the initial one.

The method proposed in Chapter 4 is also a deterministic optimization approach. It uses bisection to increase the consensus of the current solution. The key contribution is to reformulate the optimization problem of each bisection iteration into a *bi-convex program*, which can be efficiently solved by standard bi-convex optimization. In this way, we can apply efficient and deterministic optimization for consensus maximization *without relaxing the original objective function*. Therefore, no smoothing parameter tuning is required.

2.2 Globally optimal algorithms

While efficient, the algorithms mentioned in previous sections are all sub-optimal, i.e., they cannot guarantee to find the best solution possible. To address this problem, several globally optimal algorithms are proposed.

2.2.1 BnB

BnB [13] is a standard global optimization technique. It is essentially an exhaustive search algorithm. In the context of consensus maximization [24, 31, 41], BnB iteratively divides the parameter space of the model $\boldsymbol{\theta}$, and calculates the upper and lower bounds of the maximum consensus value $f_{\text{MC}}(\boldsymbol{\theta}^*|\mathcal{S})$ on all sub-spaces. The global upper and lower bounds are maintained as the maximum over all sub-spaces. As each sub-space becomes smaller and smaller, the global upper bound will gradually decrease, and the global lower bound will gradually increase. The algorithm terminates when the global upper and lower bounds meet (or are closer than a predefined threshold), and returns the solution that achieves the upper bound, i.e., the maximum consensus value. The core of BnB lies in the design of bounding functions, i.e., the way to compute upper and lower bounds. The speed of BnB depends heavily on the tightness of bounding functions. BnB has two drawbacks. One is its high computational complexity, which is worst case exponential to the size of the parameter space. The second one is the difficulty in designing the bounding functions. The form of bounding functions is highly problem-dependent, and not always trivial to construct.

Section 4.2 of Chapter 6 provides a concrete example of how to design a BnB algorithm for translation search in the problem of point cloud registration.

2.2.2 Basis enumeration

Basis enumeration [30, 15] is another type of globally optimal algorithms, which as its name suggests, enumerates all possible bases (see [30] or Chapter 5 for the definition), and fits models on them. Each basis is a subset of the input data $\mathcal{S}$ that has p data points, where $p \ll |\mathcal{S}|$. In many computer vision applications, p is only slightly larger than the dimension d of the model, e.g., $p = d + 1$. Though the complexity of basis enumeration is not related to the size of the parameter space, the number of all possible bases is $\binom{|\mathcal{S}|}{p}$, which still scales poorly with $|\mathcal{S}|$ and p.

2.2.3 Tree search

Consensus maximization can also be solved globally via tree search [25, 9, 10]. The key idea is to fit the problem into the framework of the LP-type methods [28, 37], where each iteration of the algorithm solves an LP-type problem (defined in Chapter 5). As the pioneer in this direction, Li et al. [25] utilizes Breadth-First Tree Search (BFTS) in the application of triangulation. Later, by designing more effective heuristics to guide tree search and conduct branch pruning, an A*-tree search algorithm is proposed [9, 10], which is much faster than BFTS. In practice, A* tree search has been demonstrated to be one of the most efficient globally optimal algorithms.

In addition to the empirial efficiency, a key difference between tree search and other types of globally optimal algorithms is that, tree search is *Fixed-Parameter Tractable* (FPT) [6] w.r.t. the number of outliers o (see Chapter 3 for the proof). To be more precise, the time complexity for tree search is $\mathcal{O}(d^o\text{poly}(|\mathcal{S}|, d))$ ($\text{poly}(\cdot)$ means a polynomial function), i.e., the runtime of tree search is *exponential only in o and d*. Hence, for applications where o and d is not large, tree search is theoretically still tractable even though $|\mathcal{S}|$ is large.

Nonetheless, for challenging data, previous tree search algorithms can still be inefficient for moderate $o(\geq 10)$ and $d(\geq 6)$ (see Chapter 5 for more details). To further improve the practicality of tree search, Chapter 5 proposes novel techniques that can significantly accelerate A* tree search, making it able to handle a much larger number of outliers.

2.2.4 Preprocessing

Though capable of guaranteeing the solution quality, globally optimal algorithms scale poorly with the size of the input data $|\mathcal{S}|$ and/or the number of outliers o. This is due to the theoretical hardness of consensus maximization, as shown in Chapter 3.

An interesting research direction that aims to accelerate global optimization is the development of preprocessing algorithms [31, 8], where heuristics are designed to efficiently identify/remove some true outliers *before performing global optimization*. Since these algorithm guarantees to remove only true outliers, they can reduce the input data size/outlier rates without affecting the optimal solution. Though there is no theoretical guarantee, in practice, these algorithms can often remove a considerable amount of outliers in a short time, thus significantly increasing the speed of global optimization. Inspired by this line of work, Chapter 6 applies preprocessing to accelerate BnB in the application of point cloud registration.

Chapter 3

Robust Fitting in Computer Vision: Easy or Hard?

The work contained in this chapter has been published as the following papers

Tat-Jun Chin, **Zhipeng Cai**, Frank Neumann: Robust fitting in computer vision: easy or hard? European Conference on Computer Vision (ECCV) 2018.

Tat-Jun Chin, **Zhipeng Cai**, Frank Neumann: Robust fitting in computer vision: easy or hard? International Journal on Computer Vision (IJCV) 2019.

Statement of Authorship

Title of Paper	Robust Fitting in Computer Vision: Easy or Hard?
Publication Status	☑ Published ☐ Accepted for Publication ☐ Submitted for Publication ☐ Unpublished and Unsubmitted work written in manuscript style
Publication Details	1. Chin, Tat-Jun, Zhipeng Cai, and Frank Neumann. "Robust fitting in computer vision: Easy or hard?." Proceedings of the European Conference on Computer Vision (ECCV). 2018. 2. Chin, Tat-Jun, Zhipeng Cai, and Frank Neumann. "Robust Fitting in Computer Vision: Easy or Hard?." International Journal of Computer Vision (2019): 1-13.

Principal Author

Name of Principal Author	Tat-Jun Chin		
Contribution to the Paper	Proposing the main idea and planning for the experiments. Proving the NP-hardness result, guiding the development of W[1]-hardness and APX-hardness proofs. Paper writing.		
Signature		Date	15/2/2020

Co-Author Contributions

By signing the Statement of Authorship, each author certifies that:

i. the candidate's stated contribution to the publication is accurate (as detailed above);

ii. permission is granted for the candidate in include the publication in the thesis; and

iii. the sum of all co-author contributions is equal to 100% less the candidate's stated contribution.

Name of Co-Author (Candidate)	Zhipeng Cai		
Contribution to the Paper	Proving the W[1]-hardness and APX-hardness result. Conducting experiments. Preparing for the paper draft.		
Certification:	This paper reports on original research I conducted during the period of my Higher Degree by Research candidature and is not subject to any obligations or contractual agreements with a third party that would constrain its inclusion in this thesis.		
Overall percentage (%)	40%		
Signature		Date	16/2/2020

Name of Co-Author	Frank Neumann		
Contribution to the Paper	Verifying the claims and proofs in theoretical hardness.		
Signature		Date	16/2/2020

International Journal of Computer Vision
https://doi.org/10.1007/s11263-019-01207-y

Robust Fitting in Computer Vision: Easy or Hard?

Tat-Jun Chin[1] · **Zhipeng Cai**[1] · **Frank Neumann**[1]

Received: 29 January 2019 / Accepted: 31 July 2019

Abstract
Robust model fitting plays a vital role in computer vision, and research into algorithms for robust fitting continues to be active. Arguably the most popular paradigm for robust fitting in computer vision is *consensus maximisation*, which strives to find the model parameters that maximise the number of inliers. Despite the significant developments in algorithms for consensus maximisation, there has been a lack of fundamental analysis of the problem in the computer vision literature. In particular, whether consensus maximisation is "tractable" remains a question that has not been rigorously dealt with, thus making it difficult to assess and compare the performance of proposed algorithms, relative to what is theoretically achievable. To shed light on these issues, we present several computational hardness results for consensus maximisation. Our results underline the fundamental intractability of the problem, and resolve several ambiguities existing in the literature.

Keywords Robust fitting · Consensus maximisation · Inlier set maximisation · Computational hardness

1 Introduction

Robustly fitting a geometric model onto noisy and outlier-contaminated data is a necessary capability in computer vision (Meer 2004), due to the imperfectness of data acquisition systems and preprocessing algorithms (e.g., edge detection, keypoint detection and matching). Without robustness against outliers, the estimated model will be biased, leading to failure in the overall pipeline.

In computer vision, robust fitting is typically performed under the framework of *inlier set maximisation*, a.k.a. *consensus maximisation* (Fischler and Bolles 1981), where one seeks the model with the most number of inliers. For concreteness, say we wish to estimate the parameter vector $\mathbf{x} \in \mathbb{R}^d$ that defines the linear relationship $\mathbf{a}^T\mathbf{x} = b$ from a set of outlier-contaminated measurements $\mathcal{D} = \{(\mathbf{a}_i, b_i)\}_{i=1}^N$. The consensus maximisation formulation for this problem is as follows.

Problem 1 *[MAXCON]* Given input data $\mathcal{D} = \{(\mathbf{a}_i, b_i)\}_{i=1}^N$, where $\mathbf{a}_i \in \mathbb{R}^d$ and $b_i \in \mathbb{R}$, and an inlier threshold $\epsilon \in \mathbb{R}_+$, find the $\mathbf{x} \in \mathbb{R}^d$ that maximises

$$\Psi_\epsilon(\mathbf{x} \mid \mathcal{D}) = \sum_{i=1}^{N} \mathbb{I}\left(|\mathbf{a}_i^T\mathbf{x} - b_i| \leq \epsilon\right), \tag{1}$$

where $\mathbb{I}$ returns 1 if its input predicate is true, and 0 otherwise.

The quantity $|\mathbf{a}_i^T\mathbf{x} - b_i|$ is the *residual* of the i-th measurement with respect to $\mathbf{x}$, and the value given by $\Psi_\epsilon(\mathbf{x} \mid \mathcal{D})$ is the *consensus* of $\mathbf{x}$ with respect to $\mathcal{D}$. Intuitively, the consensus of $\mathbf{x}$ is the number of inliers of $\mathbf{x}$. For the robust estimate to fit the inlier structure well, the inlier threshold ϵ must be set to an appropriate value; the large number of applications that employ the consensus maximisation framework indicate that this is usually not an obstacle.

Developing algorithms for robust fitting, specifically for consensus maximisation, is an active research area in computer vision. Currently, the most popular algorithms belong to the class of randomised sampling techniques, i.e., RANSAC (Fischler and Bolles 1981) and its variants (Choi et al. 2009; Raguram et al. 2013). Unfortunately, such techniques do not provide certainty of finding satisfactory solutions, let alone optimal ones (Tran et al. 2014).

Increasingly, attention is given to constructing *globally optimal* algorithms for robust fitting, e.g., Li (2009), Zheng et al. (2011), Enqvist et al. (2012), Bazin et al. (2013), Yang et al. (2014), Parra Bustos et al. (2014), Enqvist et al. (2015), Chin et al. (2015) and Campbell et al. (2017). Such algo-

Communicated by Yair Weiss.

Tat-Jun Chin
tat-jun.chin@adelaide.edu.au

[1] The University of Adelaide, Adelaide, Australia

Published online: 09 August 2019

rithms are able to deterministically calculate the best possible solution, i.e., the model with the highest achievable consensus. This mathematical guarantee is regarded as desirable, especially in comparison to the "rough" solutions provided by random sampling heuristics.

Recent progress in globally optimal algorithms for consensus maximisation seems to suggest that global solutions can be obtained efficiently or tractably (Li 2009; Zheng et al. 2011; Enqvist et al. 2012; Bazin et al. 2013; Yang et al. 2014; Parra Bustos et al. 2014; Enqvist et al. 2015; Chin et al. 2015; Campbell et al. 2017). Moreover, decent empirical performances have been reported. This raises hopes that good alternatives to the random sampling methods are now available. However, to what extent is the problem solved? Can we expect the global algorithms to perform well in general? Are there fundamental obstacles toward efficient robust fitting algorithms? What do we even mean by "efficient"?

1.1 Our Contributions and Their Implications

Our contributions are *theoretical*. We resolve the above ambiguities in the literature, by proving the following computational hardness results. The implications of each result are also listed below.

MAXCON is NP-hard (Sect. 2).

$\Longrightarrow$ There are no algorithms that can solve MAXCON in time polynomial to the input size, which is proportional to N and d.

MAXCON is W[1]-hard in the dimension d (Sect. 3.2).

$\Longrightarrow$ There are no algorithms that can solve MAXCON in time $f(d)\text{poly}(N)$, where $f(d)$ is an arbitrary function of d, and $\text{poly}(N)$ is a polynomial of N.

MAXCON is APX-hard (Sect. 4).

$\Longrightarrow$ There are no polynomial time algorithms that can approximate MAXCON up to $(1-\delta)\psi^*$ for any known factor δ, where ψ^* is the maximum consensus.[1]

As usual, the implications of the hardness results are subject to the standard complexity assumptions P$\neq$NP (Garey 1990) and FPT$\neq$W[1] (Downey and Fellows 1999).

Our analysis indicates the "extreme" difficulty of consensus maximisation. MAXCON is not only *intractable* (by standard notions of intractability Garey 1990; Downey and Fellows 1999), the W[1]-hardness result also suggests that any global algorithm will scale exponentially in a function of d, i.e., $N^{f(d)}$. In fact, if a conjecture of Erickson et al. (2006) holds, MAXCON cannot be solved faster than N^d. Thus, the decent performances in Li (2009), Zheng et al. (2011), Enqvist et al. (2012), Bazin et al. (2013), Yang et al. (2014), Parra Bustos et al. (2014), Enqvist et al. (2015), Chin et al. (2015) and Campbell et al. (2017) are unlikely to extend to the general cases in practical settings, where $N \geq 1000$ and $d \geq 6$ are common. More pessimistically, APX-hardness shows that MAXCON is impossible to approximate, in that there are no polynomial time approximation schemes (PTAS) (Vazirani 2001) for MAXCON.[2]

A slightly positive result is as follows.

MAXCON is FPT (fixed parameter tractable) in the number of outliers o and dimension d (Sect. 3.3).

This is achieved by applying a special case of the algorithm of Chin et al. (2015) on MAXCON to yield a runtime of $\mathcal{O}((o+1)^{d+1}\text{poly}(N,d))$. However, for most computer vision problems, the values of o and d are moderate to large (e.g., o in the range of hundreds, $d \geq 5$), hence, in practice, the FPT algorithm is fast usually only for instances where o is small (e.g., $o \leq 10$).

While this paper can also find relevance with a theoretical computer science audience, our work is important to the computer vision community since it helps to clarify the ambiguities on the efficiency and solvability of consensus maximisation (see also Sect. 1.2). Second, our analysis shows how the computational effort scales with the different input size parameters, thus suggesting more cogent ways for algorithm designers in computer vision to test and compare algorithms. Third, since developing algorithms for consensus maximisation is an active topic in computer vision, it is important for researchers to be aware of the fundamental limitations of solving the problem. Our theoretical findings also encourage researchers to consider alternative paradigms for robust fitting, e.g., deterministically convergent heuristic algorithms (Le et al. 2017; Purkait et al. 2017; Cai et al. 2018) or preprocessing techniques (Svärm et al. 2014; Parra Bustos and Chin 2015; Chin et al. 2016).

While our results are based specifically on MAXCON, which is concerned with fitting linear models, in practice, computer vision applications require the fitting of non-linear geometric models (e.g., fundamental matrix, planar perspective transforms, rotation matrices). However, while a case-by-case treatment is ideal, it is unlikely that non-linear consensus maximisation will be easier than linear consensus maximisation (Johnson et al. 1978; Ben-David et al. 2002; Aronov and Har-Peled 2008).

Note also that our purpose here is not to promote consensus maximisation as the "best" robust criterion. However, as a robust formulation that is "native" to computer vision, consensus maximisation enjoys prevalent use in the community.

[1] It may be the case that MAXCON is in class APX, i.e., that it could be approximated in polynomial time to *some* factor. However, we are not aware of any such algorithms.

[2] Since RANSAC does not provide any approximation guarantees, it is not an "approximation scheme" by standard definition (Vazirani 2001).

It is thus vital to know what is and isn't possible according to current algorithmic thinking. Second, it is unlikely that other robust criteria are easier to solve (Bernholt 2005). Although some that use differentiable robust loss functions (e.g., M-estimators) can be solved up to local optimality, it is unknown how far the local optima deviate from the global solution.

1.2 Previous Complexity Analyses of Robust Fitting

In the broader literature, complexity results have been obtained for a number of robust criteria (Bernholt 2005; Erickson et al. 2006), such as Least Median Squares (LMS), Least Quantile of Squares (LQS), and Least Trimmed Squares (LTS). However, the previous works have not studied consensus maximisation, which, as alluded to in Sect. 1.1, is of significant importance to computer vision. Moreover, the analyses in Bernholt (2005) and Erickson et al. (2006) have focussed mainly on NP-hardness, while our work here establishes a broader set of intractability results (parametrised intractability, inapproximability).

A closely related combinatorial problem is maximum feasible subsystem (MaxFS), which aims to find the largest feasible subset of a set of infeasible linear constraints. A number of complexity results have been developed by Amaldi and Kann (1995) for MaxFS. Unfortunately, we were not able to transfer the results in Amaldi and Kann (1995) to MAXCON,[3] hence, necessitating the present work.

In computer vision, a significant step towards complexity analysis of robust fitting (including consensus maximisation) was the work by Enqvist et al. (2012, 2015). Specifically, an $\mathcal{O}(N^{d+1})$ algorithm was presented, which was regarded as tractable since d is restricted to a few small values in the applications considered. Strictly speaking, however, Enqvist et al. (2012, 2015) have only established that robust fitting is in class XP (slice-wise polynomial) when parametrised by the dimension d, and this does not (yet) established tractability by standard definition (Downey and Fellows 1999). Unfortunately, our W[1]-hardness results in Sect. 3.2 rules out tractability when parametrised by d alone.[4]

[3] A dicussion between Tat-Jun Chin and Komei Fukuda in Feb 2018 suggested that a direct reduction from MaxFS to MAXCON is itself infeasible, due to the intractability of bounding a hyperplane arrangement (Fukuda et al. 1997, Theorem 5.1).

[4] If one was ever interested in only, say, robust affine registration of 2D point sets, then one could say that *robust 2D affine registration* is tractable, since the technique of Enqvist et al. (2012, 2015) can be used to construct an $\mathcal{O}(N^7)$ algorithm, which is polynomial in the number of input correspondences N. However, robust fitting in general, where N and d can both vary, is not tractable by the reasons already alluded to above.

1.3 Differences to the Conference Version

This paper is an extension of the conference version (Chin et al. 2018). The main differences to the conference version are:

- A correction is made to Algorithm 1 in Chin et al. (2018) to ensure consistency with the FPT result. Briefly, the previous Algorithm 1 conducts a depth-first tree search, whereas FPT requires breadth-first tree search.
- The runtime complexity of Algorithm 1 is corrected to $\mathcal{O}((d+1)^o \text{poly}(N,d))$. In Chin et al. (2018), it was shown as $\mathcal{O}(d^o \text{poly}(N,d))$. Note that this modification does not change the FPT outcome.
- A faster FPT algorithm (Algorithm 2) with $\mathcal{O}((o+1)^{d+1}\text{poly}(N,d))$ runtime is developed using the repeated basis checking technique of Chin et al. (2015).
- The concept of *kernelisation* (Downey and Fellows 1999) is explored for MAXCON (Sect. 3.5).
- Empirical validation of the FPT runtime bound is now provided. The performance of the FPT algorithm on real data is also investigated (Sect. 3.6).

The rest of the paper is devoted to developing the above theoretical and empirical results.

2 NP-Hrdness

The decision version of MAXCON is as follows.

Problem 2 *[MAXCON-D]* Given data $\mathcal{D} = \{(\mathbf{a}_i, b_i)\}_{i=1}^N$, an inlier threshold $\epsilon \in \mathbb{R}_+$, and a number $\psi \in \mathbb{N}_+$, does there exist $\mathbf{x} \in \mathbb{R}^d$ such that $\Psi_\epsilon(\mathbf{x} \mid \mathcal{D}) \geq \psi$?

Another well-known robust fitting paradigm is least median squares (LMS), where we seek the vector $\mathbf{x}$ that minimises the median of the residuals

$$\min_{\mathbf{x}\in\mathbb{R}^d} \quad \text{med}\left(|\mathbf{a}_1^T\mathbf{x} - b_1|, \ldots, |\mathbf{a}_N^T\mathbf{x} - b_N|\right). \tag{2}$$

LMS can be generalised by minimising the k-th largest residual instead

$$\min_{\mathbf{x}\in\mathbb{R}^d} \quad \text{kos}\left(|\mathbf{a}_1^T\mathbf{x} - b_1|, \ldots, |\mathbf{a}_N^T\mathbf{x} - b_N|\right), \tag{3}$$

where function kos returns its k-th largest input value.

Geometrically, LMS seeks the *slab* of the *smallest width* that contains *half* of the data points $\mathcal{D}$ in $\mathbb{R}^{d+1}$. A slab in $\mathbb{R}^{d+1}$ is defined by a normal vector $\mathbf{x}$ and width w as

$$h_w(\mathbf{x}) = \left\{ (\mathbf{a}, b) \in \mathbb{R}^{d+1} \;\middle|\; |\mathbf{a}^T\mathbf{x} - b| \leq \frac{1}{2}w \right\}. \tag{4}$$

Problem (3) thus seeks the thinnest slab that contains k of the points. The decision version of (3) is as follows.

Problem 3 *[k-SLAB]* Given data $\mathcal{D} = \{(\mathbf{a}_i, b_i)\}_{i=1}^N$, an integer k where $1 \leq k \leq N$, and a number $w' \in \mathbb{R}_+$, does there exist $\mathbf{x} \in \mathbb{R}^d$ such that k of the members of $\mathcal{D}$ are contained in a slab $h_w(\mathbf{x})$ of width at most w'?

k-SLAB has been proven to be NP-complete in Erickson et al. (2006).

Theorem 1 *MAXCON-D is NP-complete.*

Proof Let $\mathcal{D}$, k and w' define an instance of k-SLAB. This can be reduced to an instance of MAXCON-D by simply reusing the same $\mathcal{D}$, and setting $\epsilon = \frac{1}{2}w'$ and $\psi = k$. If the answer to k-SLAB is positive, then there is an $\mathbf{x}$ such that k points from $\mathcal{D}$ lie within vertical distance of $\frac{1}{2}w'$ from the hyperplane defined by $\mathbf{x}$, hence $\Psi_\epsilon(\mathbf{x} \mid \mathcal{D})$ must be at least ψ and the answer to MAXCON-D is also positive. Conversely, if the answer to MAXCON-D is positive, then there is an $\mathbf{x}$ such that ψ points have vertical distance of less than ϵ to $\mathbf{x}$, hence a slab that is centred at $\mathbf{x}$ of width at most w' can enclose k of the points, and the answer to k-SLAB is also positive. □

The NP-completeness of MAXCON-D implies the NP-hardness of the optimisation version MAXCON. See Sect. 1.1 for the implications of NP-hardness.

3 Parametrised Complexity

Parametrised complexity is a branch of algorithmics that investigates the inherent difficulty of problems with respect to structural parameters in the input Downey and Fellows (1999). In this section, we report several parametrised complexity results of MAXCON.

First, the *consensus set* $\mathcal{C}_\epsilon(\mathbf{x} \mid \mathcal{D})$ of $\mathbf{x}$ is defined as

$$\mathcal{C}_\epsilon(\mathbf{x} \mid \mathcal{D}) := \{i \in \{1, \ldots, N\} \mid |\mathbf{a}_i^T \mathbf{x} - b_i| \leq \epsilon\}. \tag{5}$$

An equivalent definition of consensus (1) is thus

$$\Psi_\epsilon(\mathbf{x} \mid \mathcal{D}) = |\mathcal{C}_\epsilon(\mathbf{x} \mid \mathcal{D})|. \tag{6}$$

Henceforth, we do not distinguish between the integer subset $\mathcal{C} \subseteq \{1, \ldots, N\}$ that indexes a subset of $\mathcal{D}$, and the actual data that are indexed by $\mathcal{C}$.

3.1 XP in the Dimension

The following is the *Chebyshev approximation* problem (Cheney 1966, Chapter 2) defined on the input data indexed by $\mathcal{C}$:

$$\min_{\mathbf{x} \in \mathbb{R}^d} \max_{i \in \mathcal{C}} |\mathbf{a}_i^T \mathbf{x} - b_i| \tag{7}$$

Problem (7) has the linear programming (LP) formulation

$$\begin{aligned} \min_{\mathbf{x} \in \mathbb{R}^d, \gamma \in \mathbb{R}} \quad & \gamma \\ \text{s.t.} \quad & |\mathbf{a}_i^T \mathbf{x} - b_i| \leq \gamma, \ i \in \mathcal{C}, \end{aligned} \tag{LP[$\mathcal{C}$]}$$

which can be solved in polynomial time. Chebyshev approximation also has the following property.

Lemma 1 *There is a subset $\mathcal{B}$ of $\mathcal{C}$, where $|\mathcal{B}| \leq d + 1$, such that*

$$\min_{\mathbf{x} \in \mathbb{R}^d} \max_{i \in \mathcal{B}} r_i(\mathbf{x}) = \min_{\mathbf{x} \in \mathbb{R}^d} \max_{i \in \mathcal{C}} r_i(\mathbf{x}) \tag{8}$$

Proof See Cheney (1966, Section 2.3). □

We call $\mathcal{B}$ a *basis* of $\mathcal{C}$. Mathematically, $\mathcal{B}$ is the set of active constraints to LP[$\mathcal{C}$], hence bases can be computed easily. In fact, LP[$\mathcal{B}$] and LP[$\mathcal{C}$] have the same minimisers. Further, for any subset $\mathcal{B}$ of size $d + 1$, a method by de la Vallée-Poussin can solve LP[$\mathcal{B}$] analytically in time polynomial to d; see Cheney (1966, Chapter 2) for details.

Let $\mathbf{x}$ be an arbitrary candidate solution to MAXCON, and $(\hat{\mathbf{x}}, \hat{\gamma})$ be the minimisers to LP[$\mathcal{C}_\epsilon(\mathbf{x} \mid \mathcal{D})$], i.e., the Chebyshev approximation problem on the consensus set of $\mathbf{x}$. The following property can be established.

Lemma 2 $\Psi_\epsilon(\hat{\mathbf{x}} \mid \mathcal{D}) \geq \Psi_\epsilon(\mathbf{x} \mid \mathcal{D})$.

Proof By construction, $\hat{\gamma} \leq \epsilon$. Hence, if $(\mathbf{a}_i, b_i)$ is an inlier to $\mathbf{x}$, i.e., $|\mathbf{a}_i^T \mathbf{x} - b_i| \leq \epsilon$, then $|\mathbf{a}_i^T \hat{\mathbf{x}} - b_i| \leq \hat{\gamma} \leq \epsilon$, i.e., $(\mathbf{a}_i, b_i)$ is also an inlier to $\hat{\mathbf{x}}$. Thus, the consensus of $\hat{\mathbf{x}}$ is no smaller than the consensus of $\mathbf{x}$. □

Lemmas 1 and 2 suggest a rudimentary algorithm for consensus maximisation that attempts to find the basis of the maximum consensus set, as encapsulated in the proof of the following theorem.

Theorem 2 *MAXCON is XP (slice-wise polynomial) in the dimension d.*

Proof Let $\mathbf{x}^*$ be a witness to an instance of MAXCON-D with positive answer, i.e., $\Psi_\epsilon(\mathbf{x}^* \mid \mathcal{D}) \geq \psi$. Let $(\hat{\mathbf{x}}^*, \hat{\gamma}^*)$ be the minimisers to LP[$\mathcal{C}_\epsilon(\mathbf{x}^* \mid \mathcal{D})$]. By Lemma 2, $\hat{\mathbf{x}}^*$ is also a positive witness to the instance. By Lemma 1, $\hat{\mathbf{x}}^*$ can be found by enumerating all $(d + 1)$-subsets of $\mathcal{D}$, and solving Chebyshev approximation (7) on each $(d + 1)$-subset. There are a total of $\binom{N}{d+1}$ subsets to check; including the time to evaluate $\Psi_\epsilon(\mathbf{x} \mid \mathcal{D})$ for each candidate, the runtime of this simple algorithm is $\mathcal{O}(N^{d+2}\text{poly}(d))$, which is polynomial in N for a fixed d. □

Theorem 2 shows that for a fixed dimension d, MAXCON can be solved in time polynomial in the number of measurements N [this is consistent with the results in Enqvist et al. (2012, 2015)]. However, this does not imply that MAXCON is tractable (following the standard meaning of tractability in complexity theory Garey 1990; Downey and Fellows 1999). Moreover, in practical applications, d could be large (e.g., $d \geq 5$), thus the rudimentary algorithm above will not be efficient for large N.

3.2 W[1]-Hard in the Dimension

Can we remove d from the exponent of the runtime of a globally optimal algorithm? By establishing W[1]-hardness in the dimension, this section shows that it is not possible. Our proofs are inspired by, but extends quite significantly from, that of Giannopoulos et al. (2009, Section 5). First, the source problem is as follows.

Problem 4 [k-CLIQUE] Given undirected graph $G = (V, E)$ with vertex set V and edge set E and a parameter $k \in \mathbb{N}_+$, does there exist a clique in G with k vertices?

k-CLIQUE is W[1]-hard w.r.t. parameter k[5]. Here, we demonstrate an FPT reduction from k-CLIQUE to MAXCON-D with fixed dimension d.

3.2.1 Generating the Input Data

Given input graph $G = (V, E)$, where $V = \{1, \ldots, M\}$, and size k, we construct a $(k+1)$-dimensional point set $\mathcal{D}_G = \{(\mathbf{a}_i, b_i)\}_{i=1}^{N} = \mathcal{D}_V \cup \mathcal{D}_E$ as follows:

– The set $\mathcal{D}_V$ is defined as

$$\mathcal{D}_V = \{(\mathbf{a}_\alpha^v, b_\alpha^v)\}_{\alpha=1,\ldots,k}^{v=1,\ldots,M}, \tag{9}$$

where

$$\mathbf{a}_\alpha^v = [0, \ldots, 0, 1, 0, \ldots, 0]^T \tag{10}$$

is a k-dimensional vector of 0's except at the α-th element where the value is 1, and

$$b_\alpha^v = v. \tag{11}$$

[5] https://en.wikipedia.org/wiki/Parameterized_complexity.

– The set $\mathcal{D}_E$ is defined as

$$\begin{aligned} \mathcal{D}_E = \Big\{ & (\mathbf{a}_{\alpha,\beta}^{u,v}, b_{\alpha,\beta}^{u,v}) \mid u, v = 1, \ldots, M, \\ & \langle u, v \rangle \in E, \langle v, u \rangle \in E, \\ & \alpha, \beta = 1, \ldots, k, \\ & \alpha < \beta \Big\}, \end{aligned} \tag{12}$$

where

$$\mathbf{a}_{\alpha,\beta}^{u,v} = [0, \ldots, 0, 1, 0, \ldots, 0, M, 0, \ldots, 0]^T \tag{13}$$

is a k-dimensional vector of 0's, except at the α-th element where the value is 1 and the β-th element where the value is M, and

$$b_{\alpha,\beta}^{u,v} = u + Mv. \tag{14}$$

The size N of $\mathcal{D}_G$ is thus $|\mathcal{D}_V| + |\mathcal{D}_E| = kM + 2|E|\binom{k}{2}$.

3.2.2 Setting the Inlier Threshold

Under our reduction, $\mathbf{x} \in \mathbb{R}^d$ is responsible for "selecting" a subset of the vertices V and edges E of G. First, we say that $\mathbf{x}$ selects vertex v if a point $(\mathbf{a}_\alpha^v, b_\alpha^v) \in \mathcal{D}_V$, for some α, is an inlier to $\mathbf{x}$, i.e., if

$$|(\mathbf{a}_\alpha^v)^T \mathbf{x} - b_\alpha^v| \leq \epsilon \equiv x_\alpha \in [v - \epsilon, v + \epsilon], \tag{15}$$

where x_α is the α-th element of $\mathbf{x}$. The key question is how to set the value of the inlier threshold ϵ, such that $\mathbf{x}$ selects no more than k vertices, or equivalently, such that $\Psi_\epsilon(\mathbf{x} \mid \mathcal{D}_V) \leq k$ for all $\mathbf{x}$.

Lemma 3 *If $\epsilon < \frac{1}{2}$, then $\Psi_\epsilon(\mathbf{x} \mid \mathcal{D}_V) \leq k$, with equality achieved if and only if $\mathbf{x}$ selects k vertices of G.*

Proof For any u and v, the ranges $[u - \epsilon, u + \epsilon]$ and $[v - \epsilon, v + \epsilon]$ cannot overlap if $\epsilon < \frac{1}{2}$. Hence, x_α lies in at most one of the ranges, i.e., each element of $\mathbf{x}$ selects at most one of the vertices; see Fig. 1. This implies that $\Psi_\epsilon(\mathbf{x} \mid \mathcal{D}_V) \leq k$. □

Second, a point $(\mathbf{a}_{\alpha,\beta}^{u,v}, b_{\alpha,\beta}^{u,v})$ from $\mathcal{D}_E$ is an inlier to $\mathbf{x}$ if

$$|(\mathbf{a}_{\alpha,\beta}^{u,v})^T \mathbf{x} - b_{\alpha,\beta}^{u,v}| \leq \epsilon \equiv |(x_\alpha - u) + M(x_\beta - v)| \leq \epsilon. \tag{16}$$

As suggested by (16), the pairs of elements of $\mathbf{x}$ are responsible for selecting the edges of G. To prevent each element pair x_α, x_β from selecting more than one edge, or equivalently, to maintain $\Psi_\epsilon(\mathbf{x} \mid \mathcal{D}_E) \leq \binom{k}{2}$, the setting of ϵ is crucial.

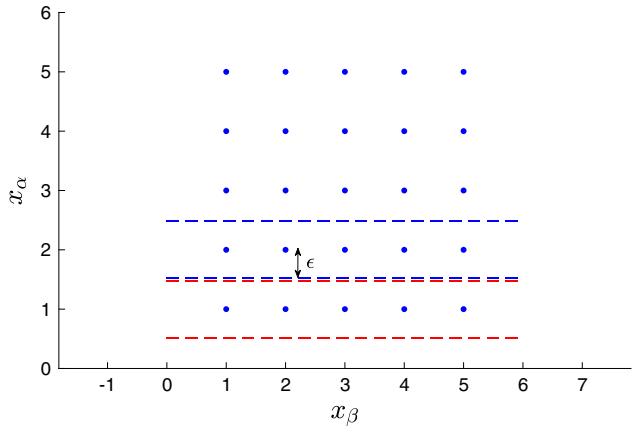


Fig. 1 The blue dots indicate the integer values in the dimensions x_α and x_β. If $\epsilon < \frac{1}{2}$, then the ranges defined by (15) for all $v = 1, \ldots, M$ do not overlap. Hence, x_α can select at most one vertex of the graph (Color figure online)

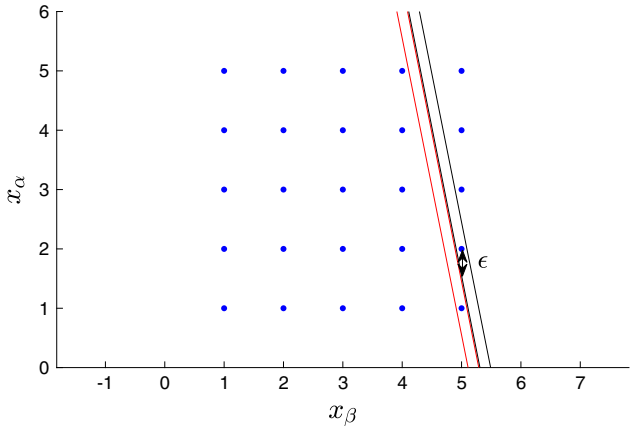


Fig. 2 The blue dots indicate the integer values in the dimensions x_α and x_β. If $\epsilon < \frac{1}{2}$, then any two slabs defined by (17) for different (u_1, v_1) and (u_2, v_2) pairs do not intersect. The figure shows two slabs corresponding to $u_1 = 1, v_1 = 5, u_2 = 2, v_2 = 5$ (Color figure online)

Lemma 4 *If $\epsilon < \frac{1}{2}$, then $\Psi_\epsilon(\mathbf{x} \mid \mathcal{D}_E) \leq \binom{k}{2}$, with equality achieved if and only if $\mathbf{x}$ selects $\binom{k}{2}$ edges of G.*

Proof For each α, β pair, the constraint (16) is equivalent to the two linear inequalities

$$\begin{aligned} x_\alpha + Mx_\beta - u - Mv &\leq \epsilon, \\ x_\alpha + Mx_\beta - u - Mv &\geq -\epsilon, \end{aligned} \tag{17}$$

which specify two opposing half-planes (i.e., a slab) in the space (x_α, x_β). Note that the slopes of the half-plane boundaries do not depend on u and v. For any two unique pairs (u_1, v_1) and (u_2, v_2), we have the four linear inequalities

$$\begin{aligned} x_\alpha + Mx_\beta - u_1 - Mv_1 &\leq \epsilon, \\ x_\alpha + Mx_\beta - u_1 - Mv_1 &\geq -\epsilon, \\ x_\alpha + Mx_\beta - u_2 - Mv_2 &\leq \epsilon, \\ x_\alpha + Mx_\beta - u_2 - Mv_2 &\geq -\epsilon. \end{aligned} \tag{18}$$

The system (18) can be simplified to

$$\begin{aligned} \frac{1}{2}\left[u_2 - u_1 + M(v_2 - v_1)\right] &\leq \epsilon, \\ \frac{1}{2}\left[u_1 - u_2 + M(v_1 - v_2)\right] &\leq \epsilon. \end{aligned} \tag{19}$$

Setting $\epsilon < \frac{1}{2}$ ensures that the two inequalities (19) cannot be consistent for all unique pairs (u_1, v_1) and (u_2, v_2). Geometrically, with $\epsilon < \frac{1}{2}$, the two slabs defined by (17) for different (u_1, v_1) and (u_2, v_2) pairs do not intersect; see Fig. 2 for an illustration.

Hence, if $\epsilon < \frac{1}{2}$, each element pair x_α, x_β of $\mathbf{x}$ can select at most one of the edges. Cumulatively, $\mathbf{x}$ can select at most $\binom{k}{2}$ edges, thus $\Psi_\epsilon(\mathbf{x} \mid \mathcal{D}_E) \leq \binom{k}{2}$. □

Up to this stage, we have shown that if $\epsilon < \frac{1}{2}$, then $\Psi_\epsilon(\mathbf{x} \mid \mathcal{D}_G) \leq k + \binom{k}{2}$, with equality achievable if there is a clique of size k in G. To establish the FPT reduction, we need to establish the reverse direction, i.e., if $\Psi_\epsilon(\mathbf{x} \mid \mathcal{D}_G) = k + \binom{k}{2}$, then there is a k-clique in G. The following lemma shows that this can be assured by setting $\epsilon < \frac{1}{M+2}$.

Lemma 5 *If $\epsilon < \frac{1}{M+2}$, then $\Psi_\epsilon(\mathbf{x} \mid \mathcal{D}_G) \leq k + \binom{k}{2}$, with equality achievable if and only if there is a clique of size k in G.*

Proof The 'only if' direction has already been proven. To prove the 'if' direction, we show that if $\epsilon < \frac{1}{M+2}$ and $\Psi_\epsilon(\mathbf{x} \mid \mathcal{D}_G) = k + \binom{k}{2}$, the subgraph $\mathrm{S}(\mathbf{x}) = \{\lfloor x_1 \rceil, \ldots, \lfloor x_k \rceil\}$ is a k-clique, where each $\lfloor x_\alpha \rceil$ represents a vertex index in G. Since $\epsilon < \frac{1}{2}$, $\lfloor x_\alpha \rceil = u$ if and only if $(\mathbf{a}^u_\alpha, b^u_\alpha)$ is an inlier. Therefore, $\mathrm{S}(\mathbf{x})$ consists of all vertices selected by $\mathbf{x}$. From Lemmas 3 and 4, when $\Psi_\epsilon(\mathbf{x} \mid \mathcal{D}_G) = k + \binom{k}{2}$, $\mathbf{x}$ is consistent with k points in $\mathcal{D}_V$ and $\binom{k}{2}$ points in $\mathcal{D}_E$. The inliers in $\mathcal{D}_V$ specifies the k vertices in $\mathrm{S}(\mathbf{x})$. The 'if' direction is true if all selected $\binom{k}{2}$ edges are only edges in $\mathrm{S}(\mathbf{x})$, i.e., for each inlier point $(\mathbf{a}^{u,v}_{\alpha,\beta}, b^{u,v}_{\alpha,\beta}) \in \mathcal{D}_E$, $(\mathbf{a}^u_\alpha, b^u_\alpha)$ and $(\mathbf{a}^v_\beta, b^v_\beta)$ are also inliers w.r.t. $\mathbf{x}$. The prove is done by contradiction:

If $\epsilon < \frac{1}{M+2}$, given an inlier $(\mathbf{a}^{u,v}_{\alpha,\beta}, b^{u,v}_{\alpha,\beta})$, from (16) we have:

$$\begin{aligned} &|(x_\alpha - u) + M(x_\beta - v)| \\ &= |[(\lfloor x_\alpha \rceil - u) + M(\lfloor x_\beta \rceil - v)] + [(x_\alpha - \lfloor x_\alpha \rceil) + M(x_\beta - \lfloor x_\beta \rceil)]| \\ &< \frac{1}{M+2}. \end{aligned} \tag{20}$$

Assume at least one of $(\mathbf{a}^u_\alpha, b^u_\alpha)$ and $(\mathbf{a}^v_\beta, b^v_\beta)$ is not an inlier, from (15) and $\epsilon < \frac{1}{M+2}$, we have $\lfloor x_\alpha \rceil \neq u$ or $\lfloor x_\beta \rceil \neq v$, which means that at least one of $(\lfloor x_\alpha \rceil - u)$ and $(\lfloor x_\beta \rceil - v)$ is not zero. Since all elements of $\mathbf{x}$ satisfy (15), both $(\lfloor x_\alpha \rceil - u)$ and $(\lfloor x_\beta \rceil - v)$ are integers between $[-(M-1), (M-1)]$.

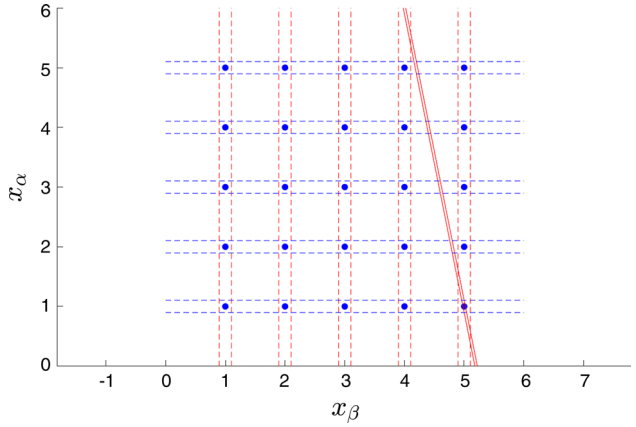


Fig. 3 If $\epsilon < \frac{1}{M+2}$, then the slab (17) that contains a point $(\mathbf{a}^{u,v}_{\alpha,\beta}, b^{u,v}_{\alpha,\beta}) \in \mathcal{D}_E$, where (u, v) is an edge in $\mathcal{G}$, does not intersect with any grid region besides the one formed by $(\mathbf{a}^u_\alpha, b^u_\alpha)$ and $(\mathbf{a}^v_\beta, b^v_\beta)$. In this figure, $u = 1$ and $v = 5$ (Color figure online)

If only one of $(\lfloor x_\alpha \rceil - u)$ and $(\lfloor x_\beta \rceil - v)$ is not zero, then $|(\lfloor x_\alpha \rceil - u) + M(\lfloor x_\beta \rceil - v)| \geq |1 + M \cdot 0| = 1$. If both are not zero, then $|(\lfloor x_\alpha \rceil - u) + M(\lfloor x_\beta \rceil - v)| \geq |(M-1) + M \cdot 1| = 1$ Therefore, we have

$$|(\lfloor x_\alpha \rceil - u) + M(\lfloor x_\beta \rceil - v)| \geq 1. \tag{21}$$

Also due to (15), we have

$$|(x_\alpha - \lfloor x_\alpha \rceil) + M(x_\beta - \lfloor x_\beta \rceil)| \leq (M+1) \cdot \epsilon = \frac{M+1}{M+2}. \tag{22}$$

Combining (21) and (22), we have

$$\begin{aligned} &|[(\lfloor x_\alpha \rceil - u) + M(\lfloor x_\beta \rceil - v)] + [(x_\alpha - \lfloor x_\alpha \rceil) + M(x_\beta - \lfloor x_\beta \rceil)]| \geq \\ &|[(\lfloor x_\alpha \rceil - u) + M(\lfloor x_\beta \rceil - v)]| - |[(x_\alpha - \lfloor x_\alpha \rceil) + M(x_\beta - \lfloor x_\beta \rceil)]| \geq \\ &1 - \frac{M+1}{M+2} = \frac{1}{M+2}, \end{aligned} \tag{23}$$

which contradicts (20). It is obvious that $\mathrm{S}(\mathbf{x})$ can be computed within linear time. Hence, the 'if' direction is true when $\epsilon < \frac{1}{M+2}$. □

To illustrate Lemma 5, Fig. 3 depicts the value of $\Psi_\epsilon(\mathbf{x} \mid \mathcal{D}_G)$ in the subspace (x_α, x_β) for $\epsilon < \frac{1}{M+2}$. Observe that $\Psi_\epsilon(\mathbf{x} \mid \mathcal{D}_G)$ attains the highest value of 3 in this subspace if and only if x_α and x_β select a pair of vertices that are connected by an edge in G.

3.2.3 Completing the Reduction

We have demonstrated a reduction from k-CLIQUE to MAXCON-D, where the main work is to generate data $\mathcal{D}_G$ which has number of measurements $N = k|V| + 2|E|\binom{k}{2}$ that is linear in $|G|$ and polynomial in k, and dimension $d = k$. In other words, the reduction is FPT in k. Setting $\epsilon < \frac{1}{M+2}$ and $\psi = k + \binom{k}{2}$ completes the reduction.

Theorem 3 *MAXCON is W[1]-hard w.r.t. the dimension d.*

Proof Since k-CLIQUE is W[1]-hard w.r.t. k, by the above FPT reduction, MAXCON is W[1]-hard w.r.t. d. □

The implications of Theorem 3 have been discussed in Sect. 1.1.

3.3 FPT in the Number of Outliers and Dimension

Let $f(\mathcal{C})$ and $\hat{\mathbf{x}}_\mathcal{C}$ respectively indicate the minimised objective value and minimiser of LP[$\mathcal{C}$]. Consider two subsets $\mathcal{P}$ and $\mathcal{Q}$ of $\mathcal{D}$, where $\mathcal{P} \subseteq \mathcal{Q}$. The statement

$$f(\mathcal{P}) \leq f(\mathcal{Q}) \tag{24}$$

follows from the fact that LP[$\mathcal{P}$] contains only a subset of the constraints of LP[$\mathcal{Q}$]; we call this property *monotonicity*.

Let $\mathbf{x}^*$ be a global solution of an instance of MAXCON, and let $\mathcal{I}^* := \mathcal{C}_\epsilon(\mathbf{x}^* \mid \mathcal{D}) \subset \mathcal{D}$ be the maximum consensus set. Let $\mathcal{C}$ index a subset of $\mathcal{D}$, and let $\mathcal{B}$ be the basis of $\mathcal{C}$. If $f(\mathcal{C}) > \epsilon$, then by Lemma 1

$$f(\mathcal{D}) \geq f(\mathcal{C}) = f(\mathcal{B}) > \epsilon. \tag{25}$$

The monotonicity property affords us further insight.

Lemma 6 *At least one point in $\mathcal{B}$ do not exist in $\mathcal{I}^*$.*

Proof By monotonicity,

$$\epsilon < f(\mathcal{B}) \leq f(\mathcal{I}^* \cup \mathcal{B}). \tag{26}$$

Hence, $\mathcal{I}^* \cup \mathcal{B}$ cannot be equal to $\mathcal{I}^*$, for if they were equal, then $f(\mathcal{I}^* \cup \mathcal{B}) = f(\mathcal{I}^*) \leq \epsilon$ which violates (26). □

The above observations suggest an algorithm for MAXCON that iteratively removes basis points to find a consensus set, as summarised in Algorithm 1. This algorithm is a special case of the technique of Chin et al. (2015). Note that in the worst case, Algorithm 1 finds a solution with consensus d (i.e., the minimal case to fit $\mathbf{x}$), if there are no solutions with higher consensus to be found.

Theorem 4 *MAXCON is FPT in the number of outliers and dimension.*

Proof Algorithm 1 conducts a breadth-first tree search to find a sequence of *basis* points to remove from $\mathcal{D}$ to yield a consensus set. By Lemma 6, the longest sequence of basis points that needs to be removed is $o = N - |\mathcal{I}^*|$, which is also the maximum tree depth searched by the algorithm (each descend

Algorithm 1 FPT algorithm for MAXCON.

Require: $\mathcal{D} = \{\mathbf{a}_i, b_i\}_{i=1}^N$, threshold ϵ.
1: $\mathcal{C}_0 \leftarrow \{1, \ldots, N\}$, $\mathbf{T} \leftarrow \{\mathcal{C}_0\}$.
2: **for** each $\mathcal{C} \in \mathbf{T}$ **do**
3: Solve LP[$\mathcal{C}$] to obtain $f(\mathcal{C})$, $\hat{\mathbf{x}}_\mathcal{C}$, and the basis $\mathcal{B}$ of $\mathcal{C}$.
4: **if** $f(\mathcal{C}) \leq \epsilon$ **then**
5: **return** $\hat{\mathbf{x}}_\mathcal{C}, \mathcal{C}$.
6: **end if**
7: **for** each $i \in \mathcal{B}$ **do**
8: Push $\{\mathcal{C} \setminus i\}$ into $\mathbf{T}$. //Remove points from basis and save the rest point set.
9: **end for**
10: **end for**

of the tree removes one point). The number of nodes visited is of order $(d+1)^o$, since for non-degenerate problem instances,[6] the branching factor of the tree is $|\mathcal{B}|$, and by Lemma 1, $|\mathcal{B}| \leq d+1$.

At each node, LP[$\mathcal{C}$] is solved, with the largest of these LPs having $d+1$ variables and N constraints. Algorithm 1 thus runs in $\mathcal{O}((d+1)^o \text{poly}(N,d))$ time, which is exponential only in the number of outliers o and dimension d. □

3.4 A Faster FPT Algorithm

In Algorithm 1, repeated bases might be traversed, which may cause redundant branches being generated during the tree search. While this does not affect the FPT outcome of Algorithm 1, it does suggest a faster FPT algorithm can be constructed of the redundant paths can be avoided.

Algorithm 2 describes the redundancy avoidance variant of Algorithm 1. Following Chin et al. (2015), the idea is to use a hash table $\mathbf{H}$ to store all the previously visited bases for efficient repetition detection. Given a basis $\mathcal{B}$, we branch (i.e., generate child bases thereof) only when it is not in $\mathbf{H}$. Theorem 5 provides the worst case runtime of this algorithm (note that while the main structure of Algorithm 2 is similar to that in Chin et al. (2015), the runtime analysis in Theorem 5 is novel).

Theorem 5 *MAXCON is solvable by Algorithm 2 in* $\mathcal{O}((o+1)^{(d+1)}\text{poly}(N,d))$ *time.*

Proof According to (Matoušek 1995, Theorem 2.3 (i)), the number of *unique* bases on and before level o is $\mathcal{O}((o+1)^{(d+1)})$. And since branching is not allowed for repeated bases, the parent basis of any repeated basis must be unique. Therefore, the number of bases traversed by Algorithm 2 is $\mathcal{O}((o+1)^{(d+1)} \cdot (d+1))$, which is the number of unique basis $\mathcal{O}((o+1)^{(d+1)})$ times the size $d+1$ of a branch. And since the runtime spent on each basis is $\text{poly}(N,d)$, the runtime of Algorithm 2 is $\mathcal{O}((o+1)^{(d+1)}) \cdot (d+1) \cdot \text{poly}(N,d) \leq \mathcal{O}((o+1)^{(d+1)}\text{poly}(N,d))$. □

[6] This can be ensured by infinitesimal data perturbations (Matoušek 1995).

Algorithm 2 FPT algorithm for MAXCON with repetition avoidance.

Require: $\mathcal{D} = \{\mathbf{a}_i, b_i\}_{i=1}^N$, threshold ϵ.
1: $\mathcal{C}_0 \leftarrow \{1, \ldots, N\}$, $\hat{\mathbf{x}} \leftarrow$ NULL, $\hat{\psi} \leftarrow 0$, $\mathbf{T} \leftarrow \{\mathcal{C}_0\}$.
2: Initialize hash table $\mathbf{H}$ to NULL.
3: **for** each $\mathcal{C} \in \mathbf{T}$ **do**
4: Solve LP[$\mathcal{C}$] to obtain $f(\mathcal{C})$, $\hat{\mathbf{x}}_\mathcal{C}$, and the basis $\mathcal{B}$ of $\mathcal{C}$.
5: **if** $f(\mathcal{C}) \leq \epsilon$ **then**
6: **return** $\hat{\mathbf{x}}_\mathcal{C}, \mathcal{C}$.
7: **end if**
8: **if** $\mathcal{B}$ does not exist in $\mathbf{H}$ **then**
9: Hash $\mathcal{B}$ into $\mathbf{H}$.
10: **for** each $i \in \mathcal{B}$ **do**
11: Push $\{\mathcal{C} \setminus i\}$ into $\mathbf{T}$.
12: **end for**
13: **end if**
14: **end for**

Note that in the context of parametrised complexity, $\mathcal{O}((o+1)^{(d+1)}) \neq \mathcal{O}((o+1)^d)$ or $\mathcal{O}(o^{(d+1)})$ since both o and d are parameters. Also, in computer vision applications, o can be much larger than d. Therefore, $\mathcal{O}((o+1)^{(d+1)})$ is generally much smaller than $\mathcal{O}((d+1)^o)$. The empirical validity and tightness of these two bounds will be shown later in Sect. 3.6.1.

3.4.1 Further Speedups with Heuristics

In Chin et al. (2015), an admissible heuristic is used to guide the tree search. This effectively changes the breadth first regime in Algorithm 2 to A*-tree search (Hart et al. 1968). While in practice, this significantly speeds up convergence, the worst case time complexity of the technique is not improved. Thus, for brevity we will not describe the A*-tree search algorithm here, and instead refer the interested reader to Chin et al. (2015). In the experiments in Sect. 3.6.1, however, we will use the A* version for practical reasons.

3.5 Relation Between FPT and Kernelisation

An important indication of a problem being FPT is the existence of the *kernel*, which is defined as follows for MAXCON following (Cygan et al. 2015, Definition 2.1).

Definition 1 A *kernelisation algorithm* for MAXCON is a polynomial time algorithm whereby given an instance of MAXCON with size $|\mathcal{I}|$, returns an equivalent instance—called a *kernel* - whose size $|\mathcal{I}'| \leq g(o,d)$ for some computable function $g : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$.

Intuitively, a kernel is a *polynomial time* algorithm that finds the "core" structure of a given problem instance whose size is bounded by a function that depends *only on the parameters* of the problem. The existence of a kernel indicates that when o and d are small (such that $g(o,d) < |\mathcal{I}|$), we might still be able to solve MAXCON efficiently. The following

lemma specifies the relationship between an FPT algorithm and the kernel.

Lemma 7 *MAXCON admits a kernel with* $g(o, d) = \mathcal{O}((o+1)^{(d+1)})$.

Proof According to the proof of Theorem 5, MAXCON can be solved in $g(o, d) \cdot \text{poly}(N, d)$ time by Algorithm 2. Assume the input size is $|\mathcal{I}|$, following the proof of Cygan et al. (2015, Lemma 2.2), we can construct the kernel by simply running Algorithm 2 and limit the maximum number of (unique) bases to traverse to $|\mathcal{I}|$.

First, since $|\mathcal{I}| = \mathcal{O}(N \cdot d)$ is the number of bits required to represent all points,[7] which is polynomial to N and d, the constructed algorithm runs in polynomial time. Meanwhile,

- **c1** If Algorithm 2 solves MAXCON before traversing $|\mathcal{I}|$ bases, the global optimal solution can be returned as the output of the kernel.
- **c2** Otherwise, we have $g(o, d) = \mathcal{O}((o+1)^{(d+1)}) \leq |\mathcal{I}|$. Hence, we can directly return the input itself as the output of the kernel since its size is already smaller than $g(o, d)$.

Therefore, the output size $|\mathcal{I}'|$ of this constructed algorithm is less than $g(o, d)$. □

Though unlike the polynomial sized *kernel* (Cygan et al. 2015, Chapter 2) (where $g(o, d)$ is polynomial and is usually derived from some heuristics), the kernel constructed from FPT algorithms cannot reduce the size of the problem, it indicates a way to test the practicality of an FPT algorithm. Namely, we can run the constructed kernel and see how often case **c1** happens in practice. If it happens for most of the problem instances, we can still hope to solve MAXCON exactly in polynomial time. The result of this test on real-world data will be demonstrated later in Sect. 3.6.2.

3.6 Experiment

All experiments in this section were executed on a laptop with Intel Core 2.60GHz i7 CPU and 16GB RAM. The code is written in MATLAB 2017a and available online.[8]

3.6.1 Empirical Tightness of the FPT Bounds

To show the validity and tightness of the two FPT bounds derived in Sect. 3.3, we test on synthetic data the difference between the number of unique nodes generated by the state-of-the-art FPT algorithm[9] (Chin et al. 2015) (called A*-tree search in the rest of this paper) and the two bounds. Specifically, the exact value of the bound for Algorithm 1 is the maximum number of (repeated) nodes traversed during tree search, which is

$$\frac{(d+1)^{(o+1)} - 1}{d} = 1 + (d+1) + \cdots + (d+1)^o \tag{27}$$

And the bound for Algorithm 2 is $e \cdot (o+1)^{(d+1)}$ (see the proof of Matoušek (1995, Theorem 2.3 (i)) for details), where e is the exponential constant.

The synthetic data was generated randomly with $d \in \{1, \ldots, 10\}$ and $o \in \{1, \ldots, 20\}$. For each fixed d and o, we generated 100 problem instances with $N = 200$. To control o, we first sampled N points from a random linear model. Then we added noise uniformly distributed in $[-\epsilon, \epsilon]$ to the b_i channel of all randomly selected inliers. Finally, outliers were created by adding truncated Gaussian noise distributed between $[-\infty, -\epsilon) \cup (\epsilon, \infty]$.

As shown in Fig. 4, the FPT bound for Algorithm 2 was much tighter than the one for Algorithm 1. Both FPT bounds were valid and the A*-tree search algorithm generated much smaller number of unique bases for all problem instances. This indicates that in practice, MAXCON can be solved much faster than the theoretical bounds.

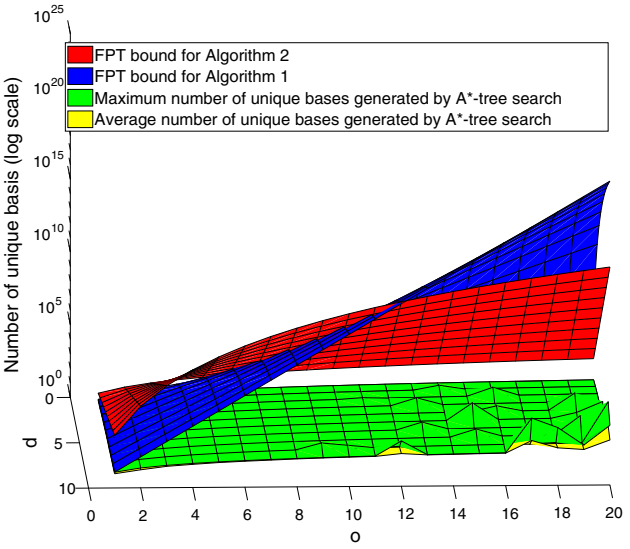
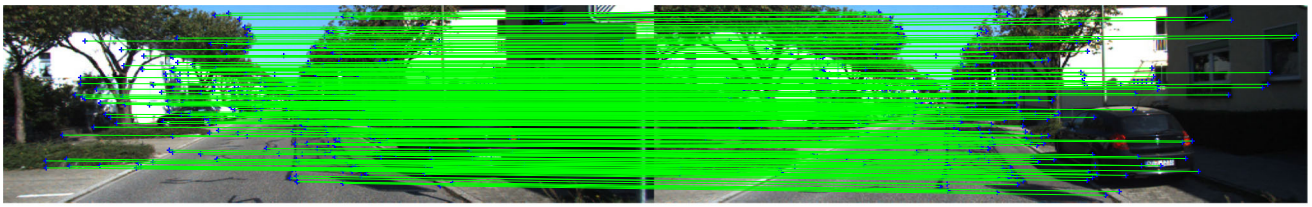


Fig. 4 Two FPT bounds and the actual number of unique bases generated by the A*-tree search. For each fixed o and d, both the maximum and the average number of unique bases generated over all 100 synthetic data are shown (Color figure online)

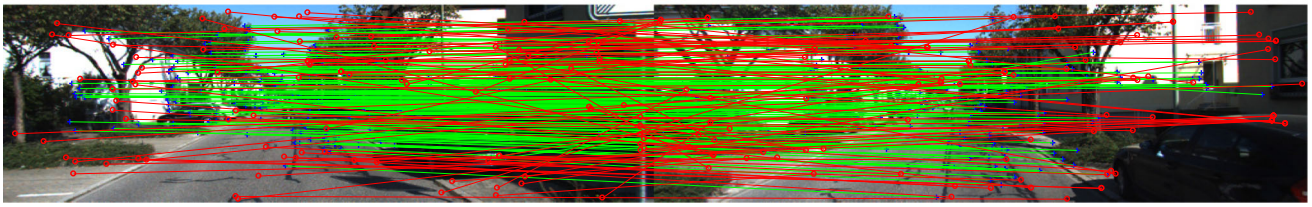
[7] https://en.wikipedia.org/wiki/Computational_complexity_theory#Measuring_the_size_of_an_instance.

[8] https://github.com/ZhipengCai/Demo---MAXCON-hardness.

[9] We do not report the result of Algorithm 1 or 2 since they are too slow even for small o and d (both algorithms did finish in 2 hours for one data instance when $d > 7$ and $o > 7$).



(a) Example data with frame gap = 1.



(b) Example data with frame gap = 5.

Fig. 5 Example inputs to the kernel of A*-tree search for linearised fundamental matrix estimation. The inliers and outliers are rendered respectively in green and red. The outliers in **b** were estimated by the sub-optimal method of Cai et al. (2018), since A*-tree search was too slow in this case (Color figure online)

3.6.2 Is Exact Solution Practical?

Since the A*-tree search was much faster than the worst case runtime, a natural question to ask is that "is it fast enough to solve MAXCON exactly in practice?" To answer this question, we used the kernel constructed in Sect. 3.5. Specifically, we ran the corresponding kernel of A*-tree search and computed the frequency that case **c1** (in the proof of Lemma 7) happened. Theoretically, the size of the problem $|\mathcal{I}|$, which was the maximum number of nodes allowed to be generated by the kernel, should be the number of bits required to represent the input data, which was $32 \cdot N \cdot d$ following the single-precision floating-point format.[10]

The experiment computed the linearised fundamental matrix (Hartley and Zisserman 2003) ($d = 8$ and the rank-2 constraint was ignored) for a sequence of image pairs from the KITTI odometry dataset (Geiger et al. 2012). For each image pair, the input to the kernel was a set of SIFT (Lowe 1999) correspondences (with N varied from 150 to 400) computed by VLFeat toolbox (Vedaldi et al. 2010). The inlier threshold was set to 0.04.

To examine the practicality of A*-tree search on image pairs with both small and large relative motions, the experiment was executed on two sequences of image pairs. In the first sequence, each image pair was made of two consecutive frames, while in the second sequence, the two images in each pair were 5 frames away. The example inputs generated from both 1-frame-gap data and 5-frame-gap data are provided in Fig. 5.

[10] https://en.wikipedia.org/wiki/Single-precision_floating-point_format.

Table 1 Practicality of A*-tree search for linearised fundamental matrix estimation. The data was generated using the images in the "00" sequence of the KITTI odometry dataset. $\bar{o}_{\max}$ and $\bar{o}_{\text{mean}}$ were respectively the mean and max of $\bar{o}$ among all data with 5 frame gap. Note that when $c1$ happened, the deepest tree level traversed by the kernel was exactly o

Frame gap	1	5
Number of tested image pairs	1000	200
Number of **c1**	1000	0
Average runtime (s) of the kernel	0.03	4340.70
Max of the deepest tree level traversed by the kernel	8	16 ($\bar{o}_{\max} = 114$)
Mean of the deepest tree level traversed by the kernel	0.04	13.15 ($\bar{o}_{\text{mean}} = 70.29$)

As shown in Table 1, the kernel terminated quickly and **c1** happened for all 1000 image pairs with 1 frame gap. This was because the relative motion was too small (see Fig. 5a for an example) and almost all SIFT correspondences were inliers ($o = 0$ for most pairs), as shown by the deepest tree level ($= o$ if **c1** happens) traversed by the kernel.

On the other hand, when the frame gap = 5 and the relative motion became large, **c1** never happened. To estimate the difference between o and the deepest tree level traversed by the kernel ($\leq$ 16 as shown in Table 1), we computed the upper bound $\bar{o}$ of o using the state-of-the-art deterministic local method (Cai et al. 2018), whose results were shown to be usually close to optimal. As shown in Table 1, the average value of $\bar{o}$, $\bar{o}_{\text{mean}} = 70.29$, was much larger than 16. Hence, the kernel was still far from finding the optimal solution when terminated. Note that, the kernel took in average more than 1 h to terminate, which was unacceptable already even though it

could find the optimal solution at the end. Therefore, solving MAXCON exactly is unlikely to be practical in general.

4 Approximability

Given the inherent intractability of MAXCON, it is natural to seek recourse in approximate solutions. However, this section shows that it is not possible to construct PTAS (Vazirani 2001) for MAXCON.

Our development here is inspired by Amaldi and Kann (1995, Sec. 3.2). First, we define our source problem: given a set of k Boolean variables $\{v_j\}_{j=1}^{k}$, a *literal* is either one of the variables, e.g., v_j, or its negation, e.g., $\neg v_j$. A *clause* is a disjunction over a set of literals, i.e., $v_1 \vee \neg v_2 \vee v_3$. A *truth assignment* is a setting of the values of the k variables. A clause is *satisfied* if it evaluates to true.

Problem 5 *[MAX-2SAT]* Given M clauses $\mathcal{K} = \{\mathcal{K}_i\}_{i=1}^{M}$ over k Boolean variables $\{v_j\}_{j=1}^{k}$, where each clause has exactly two literals, what is the maximum number of clauses that can be satisfied by a truth assignment?

MAX-2SAT is APX-hard,[11] meaning that there are no algorithms that run in polynomial time that can approximately solve MAX-2SAT up to a desired error ratio. Here, we show an L-reduction[12] from MAX-2SAT to MAXCON, which unfortunately shows that MAXCON is also APX-hard.

4.1 Generating the Input Data

Given an instance of MAX-2SAT with clauses $\mathcal{K} = \{\mathcal{K}_i\}_{i=1}^{M}$ over variables $\{v_j\}_{j=1}^{k}$, let each clause $\mathcal{K}_i$ be represented as $(\pm v_{\alpha_i}) \vee (\pm v_{\beta_i})$, where $\alpha_i, \beta_i \in \{1, \ldots, k\}$ index the variables that exist in $\mathcal{K}_i$, and $\pm$ here indicates either a "blank" (no negation) or $\neg$ (negation). Define

$$\operatorname{sgn}(\alpha_i) = \begin{cases} +1 & \text{if } v_{\alpha_i} \text{ occurs without negation in } \mathcal{K}_i, \\ -1 & \text{if } v_{\alpha_i} \text{ occurs with negation in } \mathcal{K}_i; \end{cases} \tag{28}$$

similarly for $\operatorname{sgn}(\beta_i)$. Construct the input data for MAXCON as

$$\mathcal{D}_{\mathcal{K}} = \{(\mathbf{a}_i^p, b_i^p)\}_{i=1,\ldots,M}^{p=1,\ldots,6}, \tag{29}$$

where there are six measurements for each clause. Namely, for each clause $\mathcal{K}_i$,

- $\mathbf{a}_i^1$ is a k-dimensional vector of zeros, except at the α_i-th and β_i-th elements where the values are respectively $\operatorname{sgn}(\alpha_i)$ and $\operatorname{sgn}(\beta_i)$, and $b_i^1 = 2$.
- $\mathbf{a}_i^2 = \mathbf{a}_i^1$ and $b_i^2 = 0$.
- $\mathbf{a}_i^3$ is a k-dimensional vector of zeros, except at the α_i-th element where the value is $\operatorname{sgn}(\alpha_i)$, and $b_i^3 = -1$.
- $\mathbf{a}_i^4 = \mathbf{a}_i^3$ and $b_i^4 = 1$.
- $\mathbf{a}_i^5$ is a k-dimensional vector of zeros, except at the β_i-th element where the value is $\operatorname{sgn}(\beta_i)$, and $b_i^5 = -1$.
- $\mathbf{a}_i^6 = \mathbf{a}_i^5$ and $b_i^6 = 1$.

The number of measurements N in $\mathcal{D}_{\mathcal{K}}$ is $6M$.

4.2 Setting the Inlier Threshold

Given a solution $\mathbf{x} \in \mathbb{R}^k$ for MAXCON, the six input measurements associated with $\mathcal{K}_i$ are inliers under these conditions:

$$\begin{aligned} & (\mathbf{a}_i^1, b_i^1) \text{ is an inlier} \\ \iff & |\operatorname{sgn}(\alpha_i) x_{\alpha_i} + \operatorname{sgn}(\beta_i) x_{\beta_i} - 2| \leq \epsilon, \\ & (\mathbf{a}_i^2, b_i^2) \text{ is an inlier} \\ \iff & |\operatorname{sgn}(\alpha_i) x_{\alpha_i} + \operatorname{sgn}(\beta_i) x_{\beta_i}| \leq \epsilon, \end{aligned} \tag{30}$$

$$\begin{aligned} (\mathbf{a}_i^3, b_i^3) \text{ is an inlier} &\iff |\operatorname{sgn}(\alpha_i) x_{\alpha_i} + 1| \leq \epsilon, \\ (\mathbf{a}_i^4, b_i^4) \text{ is an inlier} &\iff |\operatorname{sgn}(\alpha_i) x_{\alpha_i} - 1| \leq \epsilon, \end{aligned} \tag{31}$$

$$\begin{aligned} (\mathbf{a}_i^5, b_i^5) \text{ is an inlier} &\iff |\operatorname{sgn}(\beta_i) x_{\beta_i} + 1| \leq \epsilon, \\ (\mathbf{a}_i^6, b_i^6) \text{ is an inlier} &\iff |\operatorname{sgn}(\beta_i) x_{\beta_i} - 1| \leq \epsilon, \end{aligned} \tag{32}$$

where x_α is the α-th element of $\mathbf{x}$. Observe that if $\epsilon < 1$, then at most one of (30), one of (31), and one of (32) can be satisfied. The following result establishes an important condition for L-reduction.

Lemma 8 *If $\epsilon < 1$, then*

$$\text{OPT}(\textit{MAXCON}) \leq 6 \cdot \text{OPT}(\textit{MAX-2SAT}), \tag{33}$$

OPT(*MAX-2SAT*) *is the maximum number of clauses that can be satisfied for a given MAX-2SAT instance, and* OPT(*MAXCON*) *is the maximum achievable consensus for the MAXCON instance generated under our reduction.*

Proof If $\epsilon < 1$, for all $\mathbf{x}$, at most one of (30), one of (31), and one (32), can be satisfied, hence OPT(MAXCON) cannot be greater than $3M$. For any MAX-2SAT instance with M clauses, there is an algorithm (Johnson 1974) that can satisfy at least $\lceil \frac{M}{2} \rceil$ of the clauses, thus OPT(MAX-2SAT) $\geq \lceil \frac{M}{2} \rceil$. This leads to (33). □

Note that, if $\epsilon < 1$, rounding $\mathbf{x}$ to its nearest bipolar vector (i.e,, a vector that contains only -1 or 1) cannot decrease the

[11] https://en.wikipedia.org/wiki/2-satisfiability.

[12] https://en.wikipedia.org/wiki/L-reduction.

consensus w.r.t. $\mathcal{D}_{\mathcal{K}}$. It is thus sufficient to consider $\mathbf{x}$ that are bipolar in the rest of this section.

Intuitively, $\mathbf{x}$ is used as a proxy for truth assignment: setting $x_j = 1$ implies setting $v_j = true$, and vice versa. Further, if one of the conditions in (30) holds for a given $\mathbf{x}$, then the clause $\mathcal{K}_i$ is satisfied by the truth assignment. Hence, for $\mathbf{x}$ that is bipolar and $\epsilon < 1$,

$$\Psi_\epsilon(\mathbf{x} \mid \mathcal{D}_{\mathcal{K}}) = 2M + \sigma, \tag{34}$$

where σ is the number of clauses satisfied by $\mathbf{x}$. This leads to the final necessary condition for L-reduction.

Lemma 9 *If $\epsilon < 1$, then*

$$\begin{aligned} &|\text{OPT}(\textit{MAX-2SAT}) - \text{SAT}(\mathbf{t}(\mathbf{x}))| \\ =&|\text{OPT}(\textit{MAXCON}) - \Psi_\epsilon(\mathbf{x} \mid \mathcal{D}_{\mathcal{K}})|, \end{aligned} \tag{35}$$

where $\mathbf{t}(\mathbf{x})$ returns the truth assignment corresponding to $\mathbf{x}$, and $\text{SAT}(\mathbf{t}(\mathbf{x}))$ returns the number of clauses satisfied by $\mathbf{t}(\mathbf{x})$.

Proof For any bipolar $\mathbf{x}$ with consensus $2M + \sigma$, the truth assignment $\mathbf{t}(\mathbf{x})$ satisfies exactly σ clauses. Since the value of OPT(MAXCON) must take the form $2M + \sigma^*$, then $\text{OPT(MAX-2SAT)} = \sigma^*$. The condition (35) is immediately seen to hold by substituting the values into the equation. □

We have demonstrated an L-reduction from MAX-2SAT to MAXCON, where the main work is to generate $\mathcal{D}_{\mathcal{K}}$ in linear time. The function $\mathbf{t}$ also takes linear time to compute. Setting $\epsilon < 1$ completes the reduction.

Theorem 6 *MAXCON is APX-hard.*

Proof Since MAX-2SAT is APX-hard, by the above L-reduction, MAXCON is also APX-hard. □

See Sect. 1.1 for the implications of Theorem 6.

5 Conclusions and Future Work

Given the fundamental difficulty of consensus maximisation as implied by our results (see Sect. 1.1), it would be prudent to consider alternative paradigms for optimisation, e.g., deterministically convergent heuristic algorithms (Le et al. 2017; Purkait et al. 2017; Cai et al. 2018) or preprocessing techniques (Svärm et al. 2014; Parra Bustos and Chin 2015; Chin et al. 2016).

Acknowledgements This work was supported by ARC Grant DP160103490.

References

Amaldi, E., & Kann, V. (1995). The complexity and approximability of finding maximum feasible subsystems of linear relations. *Theoretical Computer Science*, *147*, 181–210.

Aronov, B., & Har-Peled, S. (2008). On approximating the depth and related problems. *SIAM Journal on Computing*, *38*(3), 899–921.

Bazin, J. C., Li, H., Kweon, I. S., Demonceaux, C., Vasseur, P., & Ikeuchi, K. (2013). A branch-and-bound approach to correspondence and grouping problems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *35*(7), 1565–1576.

Ben-David, S., Eiron, N., & Simon, H. (2002). The computational complexity of densest region detection. *Journal of Computer and System Sciences*, *64*(1), 22–47.

Bernholt, T. (2005). Robust estimators are hard to compute. *Technical report 52,* Technische Universität.

Cai, Z., Chin, T. J., Le, H., Suter, D. (2018). Deterministic consensus maximization with biconvex programming. In *European conference on computer vision (ECCV)*.

Campbell, D., Petersson, L., Kneip, L., Li, H. (2017). Globally-optimal inlier set maximisation for simultaneous camera pose and feature correspondence. In *IEEE international conference on computer vision (ICCV)*.

Cheney, E. W. (1966). *Introduction to approximation theory*. New York: McGraw-Hill.

Chin, T. J., Cai, Z., Neumann, F. (2018). Robust fitting in computer vision: easy or hard? In *European conference on computer vision (ECCV)*.

Chin, T. J., Kee, Y. H., Eriksson, A., Neumann, F. (2016). Guaranteed outlier removal with mixed integer linear programs. In *IEEE computer society conference on computer vision and pattern recognition (CVPR)*.

Chin, T. J., Purkait, P., Eriksson, A., Suter, D. (2015). Efficient globally optimal consensus maximisation with tree search. In *IEEE computer society conference on computer vision and pattern recognition (CVPR)*.

Choi, S., Kim, T., Yu, W. (2009). Performance evaluation of RANSAC family. In *British machine vision conference (BMVC)*.

Cygan, M., Fomin, F. V., Kowalik, Ł., Lokshtanov, D., Marx, D., Pilipczuk, M., et al. (2015). *Parameterized algorithms* (Vol. 3). Berlin: Springer.

Downey, R. G., & Fellows, M. R. (1999). *Parametrized complexity*. New York: Springer.

Enqvist, O., Ask, E., Kahl, F., & Åström, K. (2012). Robust fitting for multiple view geometry. In *European conference on computer vision (ECCV)*.

Enqvist, O., Ask, E., Kahl, F., & Åström, K. (2015). Tractable algorithms for robust model estimation. *International Journal of Computer Vision*, *112*(1), 115–129.

Erickson, J., Har-Peled, S., & Mount, D. M. (2006). On the least median square problem. *Discrete & Computational Geometry*, *36*(4), 593–607.

Fischler, M. A., & Bolles, R. C. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, *24*(6), 381–395.

Fukuda, K., Liebling, T. M., & Margot, F. (1997). Analysis of backtrack algorithms for listing all vertices and all faces of a convex polyhedron. *Computational Geometry*, *8*, 1–12.

Garey, M. R., & Johnson, D. S. (1990). *Computers and intractability: a guide to the theory of NP-completeness*. New York: W H Freeman & Co.

Geiger, A., Lenz, P., Urtasun, R. (2012). Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on computer vision and pattern recognition (CVPR)*.

Giannopoulos, P., Knauer, C., Rote, G.: The parameterized complexity of some geometric problems in unbounded dimension. In *International workshop on parameterized and exact computation (IWPEC)* (2009).

Hart, P. E., Nilsson, N. J., & Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions Systems Science and Cybernetics*, *4*(2), 100–107.

Hartley, R., & Zisserman, A. (2003). *Multiple view geometry in computer vision*. Cambridge: Cambridge University Press.

Johnson, D. S. (1974). Approximation algorithms for combinatorial problems. *Journal of Computer and System Sciences*, *9*, 256–278.

Johnson, D. S., & Preparata, F. P. (1978). The densest hemisphere problem. *Theoretical Computer Science*, *6*, 93–107.

Le, H., Chin, T. J., Suter, D. (2017). An exact penalty method for locally convergent maximum consensus. In *IEEE computer society conference on computer vision and pattern recognition (CVPR)*.

Li, H. (2009). Consensus set maximization with guaranteed global optimality for robust geometry estimation. In: *IEEE international conference on computer vision (ICCV)*.

Lowe, D. G. (1999) Object recognition from local scale-invariant features. In *The proceedings of the seventh IEEE international conference on computer vision, 1999* (Vol. 2, pp. 1150–1157). IEEE

Matoušek, J. (1995). On geometric optimization with few violated constraints. *Discrete and Computational Geometry*, *14*(4), 365–384.

Meer, P. (2004). Robust techniques for computer vision. In G. Medioni & S. B. Kang (Eds.), *Emerging topics in computer vision*. New York: Prentice Hall.

Parra Bustos, A., Chin, T. J., Suter, D.: Fast rotation search with stereographic projections for 3d registration. In *IEEE computer society conference on computer vision and pattern recognition (CVPR)* (2014)

Parra Bustos, A., Chin, T. J. (2015). Guaranteed outlier removal for rotation search. In *IEEE international conference on computer vision (ICCV)*.

Purkait, P., Zach, C., Eriksson, A. (2017) Maximum consensus parameter estimation by reweighted L1 methods. In *Energy minimization methods in computer vision and pattern recognition (EMMCVPR)*.

Raguram, R., Chum, O., Pollefeys, M., Matas, J., & Frahm, J. M. (2013). USAC: A universal framework for random sample consensus. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *35*(8), 2022–2038.

Svärm, L., Enqvist, O., Oskarsson, M., Kahl, F. (2014). Accurate localization and pose estimation for large 3d models. In *IEEE computer society conference on computer vision and pattern recognition (CVPR)*

Tran, Q. H., Chin, T. J., Chojnacki, W., & Suter, D. (2014). Sampling minimal subsets with large spans for robust estimation. *International Journal of Computer Vision (IJCV)*, *106*(1), 93–112.

Vazirani, V. (2001). *Approximation algorithms*. Berlin: Springer.

Vedaldi, A., Fulkerson, B. (2010). Vlfeat: An open and portable library of computer vision algorithms. In *Proceedings of the 18th ACM international conference on Multimedia* (pp. 1469–1472). ACM.

Yang, J., Li, H., Jia, Y.: Optimal essential matrix estimation via inlier-set maximization. In *European conference on computer vision (ECCV)*

Zheng, Y., Sugimoto, S., Okutomi, M.: Deterministically maximizing feasible subsystems for robust model fitting with unit norm constraints. In *IEEE computer society conference on computer vision and pattern recognition (CVPR)* (2011)

Chapter 4

Deterministic Consensus Maximization with Biconvex Programming

The work contained in this chapter has been published as the following papers

Zhipeng Cai, Tat-Jun Chin, Huu Le, David Suter: Deterministic consensus maximization with biconvex programming. European Conference on Computer Vision (ECCV) 2018.

Statement of Authorship

Title of Paper	Deterministic Consensus Maximization with Biconvex Programming
Publication Status	☑ Published ☐ Accepted for Publication ☐ Submitted for Publication ☐ Unpublished and Unsubmitted work written in manuscript style
Publication Details	Cai, Zhipeng, Tat-Jun Chin, Huu Le, and David Suter. "Deterministic consensus maximization with biconvex programming." In Proceedings of the European Conference on Computer Vision (ECCV), pp. 685-700. 2018.

Principal Author

Name of Principal Author (Candidate)	Zhipeng Cai		
Contribution to the Paper	Proposing the main idea. Conducting experiments. Paper writing.		
Overall percentage (%)	60%		
Certification:	This paper reports on original research I conducted during the period of my Higher Degree by Research candidature and is not subject to any obligations or contractual agreements with a third party that would constrain its inclusion in this thesis. I am the primary author of this paper.		
Signature		Date	16/2/2020

Co-Author Contributions

By signing the Statement of Authorship, each author certifies that:

i. the candidate's stated contribution to the publication is accurate (as detailed above);

ii. permission is granted for the candidate in include the publication in the thesis; and

iii. the sum of all co-author contributions is equal to 100% less the candidate's stated contribution.

Name of Co-Author	Tat-Jun Chin		
Contribution to the Paper	Providing major discussions and suggestions about the method and experiments. Modifications of the paper draft.		
Signature		Date	15/2/2020

Name of Co-Author	Huu Le		
Contribution to the Paper	Providing suggestions and source code during the experiment.		
Signature		Date	17/2/2020

Name of Co-Author	David Suter		
Contribution to the Paper	Providing suggestions about the experiments. Proofreading of the paper draft.		
Signature		Date	17/04/20

Deterministic Consensus Maximization with Biconvex Programming

Zhipeng Cai[1], Tat-Jun Chin[1], Huu Le[2], and David Suter[3]

[1] School of Computer Science, The University of Adelaide
{zhipeng.cai,tat-jun.chin}@adelaide.edu.au
[2] School of Electrical Engineering and Computer Science, Queensland University of Technology
huu.le@qut.edu.au
[3] School of Computing and Security, Edith Cowan University
d.suter@ecu.edu.au

Abstract. Consensus maximization is one of the most widely used robust fitting paradigms in computer vision, and the development of algorithms for consensus maximization is an active research topic. In this paper, we propose an efficient *deterministic optimization* algorithm for consensus maximization. Given an initial solution, our method conducts a *deterministic search* that forcibly increases the consensus of the initial solution. We show how each iteration of the update can be formulated as an instance of biconvex programming, which we solve efficiently using a novel biconvex optimization algorithm. In contrast to our algorithm, previous consensus improvement techniques rely on random sampling or relaxations of the objective function, which reduce their ability to significantly improve the initial consensus. In fact, on challenging instances, the previous techniques may even return a worse off solution. Comprehensive experiments show that our algorithm can consistently and greatly improve the quality of the initial solution, without substantial cost.[4]

Keywords: Robust fitting · Consensus maximization · Biconvex programming

1 Introduction

Due to the existence of noise and outliers in real-life data, robust model fitting is necessary to enable many computer vision applications. Arguably the most prevalent robust technique is random sample consensus (RANSAC) [11], which aims to find the model that has the largest consensus set. The RANSAC algorithm approximately solves this optimization problem, by repetitively sampling minimal subsets of the data, in the hope of "hitting" an all-inlier minimal subset that gives rise to a model hypothesis with high consensus.

[4] Matlab demo program is available at https://github.com/ZhipengCai/Demo---Deterministic-consensus-maximization-with-biconvex-programming.

Many variants of RANSAC have been proposed [7]. Most variants attempt to conduct guided sampling using various heuristics, so as to speed up the retrieval of all-inlier minimal subsets. Fundamentally, however, taking minimal subsets reduces the span of the data and produces biased model estimates [20, 27]. Thus, the best hypothesis found by RANSAC often has much lower consensus than the maximum achievable, especially on higher-dimensional problems. In reality, the RANSAC solution should only be taken as a rough initial estimate [9].

To "polish" a rough RANSAC solution, one can perform least squares (LS) on the consensus set of the RANSAC estimate (i.e. the Gold Standard Algorithm [12, Chap. 4]). Though justifiable from a maximum likelihood point of view, the efficacy of LS depends on having a sufficiently large consensus set to begin with.

A more useful approach is Locally Optimized RANSAC (LO-RANSAC) [9, 18], which attempts to enlarge the consensus set of an initial RANSAC estimate, by generating hypotheses from *larger-than-minimal subsets* of the consensus set.[5] The rationale is that hypotheses fitted on a larger number of inliers typically lead to better estimates with even higher support. Ultimately, however, LO-RANSAC is also a randomized algorithm. Although it conducts a more focused sampling, the algorithm cannot guarantee improvements to the initial estimate. As we will demonstrate in Sec. 5.2, often on more challenging datasets, LO-RANSAC is unable to significantly improve upon the RANSAC result.

Due to its combinatorial nature, consensus set maximization is NP-hard [4]. While this has not deterred the development of globally optimal algorithms [21, 30, 19, 10, 6, 5, 25, 3], the fundamental intractability of the problem means that global algorithms are essentially variants of exhaustive search-and-prune procedures, whose runtime scales exponentially in the general case. While global algorithms have their place in computer vision, currently they are mostly confined to problems with low-dimensions and/or small number of measurements.

1.1 Deterministic algorithms—a new class of methods

Recently, efficient deterministic algorithms for consensus maximization are gaining attention [17, 22]. Different from random sampling, such algorithms begin with an initial solution (obtained using least squares or a random sampling method) and iteratively performs *deterministic updates* on the solution to improve its quality. While they do not strive for the global optimum, such algorithms are able to find excellent solutions due to the directed search.

To perform deterministic updating, the previous methods relax the objective function (Le et al. [17] use ℓ_1 penalization, and Purkait et al. [22] use a smooth surrogate function). Invariably this necessitates the setting of a smoothing parameter that controls the degree of relaxation, and the progressive tightening of the relaxation to ensure convergence to a good solution. As we will demonstrate in Sec. 5.4, incorrect settings of the smoothing parameter and/or its annealing rate may actually lead to a worse solution than the starting point.

[5] This is typically invoked from within a main RANSAC routine.

1.2 Our contributions

We propose a novel deterministic optimization algorithm for consensus maximization. The overall structure of our method is a bisection search to increase the consensus of the current solution. The key to the effectiveness of our method is to formulate the feasibility test in each iteration as a *biconvex program*, which we solve efficiently via a biconvex optimization algorithm. Unlike [17, 22], our method neither relaxes the objective function, nor requires tuning of smoothing parameters. On both synthetic and real datasets, we demonstrate the superior performance of our method over previous consensus improvement techniques.

2 Problem definition

Given a set of N outlier contaminated measurements, consensus maximization aims to find the model $\mathbf{x} \in D$ that is consistent with the largest data subset

$$\underset{\mathbf{x}\in D}{\text{maximize}} \;\; \mathcal{I}(\mathbf{x}), \tag{1}$$

where D is the domain of model parameters (more details later), and

$$\mathcal{I}(\mathbf{x}) = \sum_{i=1}^{N} \mathbb{I}\left(r_i(\mathbf{x}) \leq \epsilon\right) \tag{2}$$

counts the number of inliers (consensus) of $\mathbf{x}$. Function $r_i(\mathbf{x})$ gives the *residual* of the i-th measurement w.r.t. $\mathbf{x}$, ϵ is the inlier threshold and $\mathbb{I}$ is the indicator function which returns 1 if the input statement is true and 0 otherwise.

Fig. 1 illustrates the objective function $\mathcal{I}(\mathbf{x})$. As can be appreciated from the inlier counting operations, $\mathcal{I}(\mathbf{x})$ is a step function with uninformative gradients.

2.1 The update problem

Let $\tilde{\mathbf{x}}$ be an initial solution to (1); we wish to improve $\tilde{\mathbf{x}}$ to yield a better solution. We define this task formally as

$$\text{find} \quad \mathbf{x} \in D, \quad \text{such that} \quad \mathcal{I}(\mathbf{x}) \geq \delta, \tag{3}$$

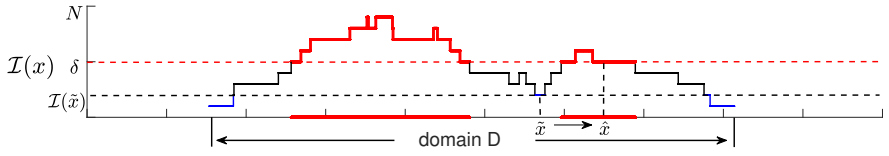


Fig. 1: Illustrating the update problem. Given the current solution $\tilde{\mathbf{x}}$ and a target consensus δ, where $\delta > \mathcal{I}(\tilde{\mathbf{x}})$, the update problem (3) aims to find another solution $\hat{\mathbf{x}}$ with $\mathcal{I}(\hat{\mathbf{x}}) \geq \delta$. Later in Sec. 4, problem (3) will be embedded in a broader algorithm that searches over δ to realize deterministic consensus maximization.

where $\delta > \mathcal{I}(\tilde{\mathbf{x}})$ is a target consensus value. See Fig. 1 for an illustration. For now, assume that δ is given; later in Sec. 4 we will embed (3) in a broader algorithm to search over δ.

Also, although (3) does not demand that the revised solution be "close" to $\tilde{\mathbf{x}}$, it is strategic to employ $\tilde{\mathbf{x}}$ as a starting point to perform the update. In Sec. 3, we will propose such an algorithm that is able to efficiently solve (3).

2.2 Residual functions and solvable models

Before embarking on a solution for (3), it is vital to first elaborate on the form of $r_i(\mathbf{x})$ and the type of models that can be fitted by the proposed algorithm. Following previous works [14, 15, 5], we focus on residual functions of the form

$$r_i(\mathbf{x}) = \frac{q_i(\mathbf{x})}{p_i(\mathbf{x})}, \tag{4}$$

where $q_i(\mathbf{x})$ is convex quadratic and $p_i(\mathbf{x})$ is linear. We also insist that $p_i(\mathbf{x})$ positive. We call $r_i(\mathbf{x})$ the *quasiconvex geometric residual* since it is quasiconvex [2, Sec. 3.4.1] in the domain

$$D = \{\mathbf{x} \in \mathbb{R}^d \mid p_i(\mathbf{x}) > 0, i = 1, \ldots, N\}, \tag{5}$$

Note that D in the above form specifies a convex domain in $\mathbb{R}^d$.

Many model fitting problems in computer vision have residuals of the type (4). For example, in multiple view triangulation where we aim to estimate the 3D point $\mathbf{x} \in \mathbb{R}^3$ from multiple (possibly incorrect) 2D observations $\{\mathbf{u}_i\}_{i=1}^N$,

$$r_i(\mathbf{x}) = \frac{\|(\mathbf{P}_i^{(1:2)} - \mathbf{u}_i\mathbf{P}_i^{(3)})\bar{\mathbf{x}}\|_2}{\mathbf{P}_i^{(3)}\bar{\mathbf{x}}} \tag{6}$$

is the reprojection error in the i-th camera, where $\bar{\mathbf{x}} = [\mathbf{x}^T\ 1]^T$,

$$\mathbf{P}_i = \begin{bmatrix} \mathbf{P}_i^{(1:2)} \\ \mathbf{P}_i^{(3)} \end{bmatrix} \in \mathbb{R}^{3\times 4} \tag{7}$$

is the i-th camera matrix with $\mathbf{P}_i^{(1:2)}$ and $\mathbf{P}_i^{(3)}$ respectively being the first-two rows and third row of $\mathbf{P}$. Insisting that $\mathbf{x}$ lies in the convex domain $D = \{\mathbf{x} \in \mathbb{R}^3 \mid \mathbf{P}_i^{(3)}\bar{\mathbf{x}} > 0, \forall i\}$ ensures that the estimated $\mathbf{x}$ lies in front of all the cameras.

Other model fitting problems with quasiconvex geometric residuals include homography fitting, camera resectioning, and the known rotation problem; see [14] for details and other examples. However, note that fundamental matrix estimation is not a quasiconvex problem [14]; in Sec. 5, we will show how the proposed technique can be adapted to robustly estimate the fundamental matrix.

3 Solving the update problem

As the decision version of (1), the update problem (3) is NP-complete [4] and thus can only be approximately solved. In this section, we propose an algorithm that works well in practice, i.e., able to significantly improve $\tilde{\mathbf{x}}$.

3.1 Reformulation as continuous optimization

With quasicovex geometric residuals (4), the inequality $r_i(\mathbf{x}) \leq \epsilon$ becomes

$$q_i(\mathbf{x}) - \epsilon p_i(\mathbf{x}) \leq 0. \tag{8}$$

Since $q_i(\mathbf{x})$ is convex and $p_i(\mathbf{x})$ is linear, the constraint (8) specifies a convex region in D. Defining

$$r'_i(\mathbf{x}) := q_i(\mathbf{x}) - \epsilon p_i(\mathbf{x}) \tag{9}$$

and introducing for each $r'_i(\mathbf{x})$ an indicator variable $y_i \in [0, 1]$ and a slack variable $s_i \geq 0$, we can write (3) using complementarity constraints [13] as

$$\begin{aligned}
&\text{find} && \mathbf{x} \in D && \text{(10a)}\\
&\text{subject to} && \sum_i y_i \geq \delta, && \text{(10b)}\\
& && y_i \in [0, 1], \ \ \forall i, && \text{(10c)}\\
& && y_i s_i = 0, \ \ \forall i, && \text{(10d)}\\
& && s_i - r'_i(\mathbf{x}) \geq 0, \ \ \forall i, && \text{(10e)}\\
& && s_i \geq 0, \ \ \forall i. && \text{(10f)}
\end{aligned}$$

Intuitively, y_i reflects whether the i-th datum is an inlier w.r.t. $\mathbf{x}$. In the following, we establish the integrality of y_i and the equivalence between (10) and (3).

Lemma 1. *Problems* (10) *and* (3) *are equivalent.*

Proof. Observe that for any $\mathbf{x}$,

a1: If $r'_i(\mathbf{x}) > 0$, the i-th datum is outlying to $\mathbf{x}$, and (10d) and (10e) will force $s_i \geq r'_i(\mathbf{x}) > 0$ and $y_i = 0$.

a2: If $r'_i(\mathbf{x}) \leq 0$, the i-th datum is inlying to $\mathbf{x}$, and (10f) and (10d) allow s_i and y_i to have only one of the following settings: **a2.1:** $s_i > 0$ and $y_i = 0$; or **a2.2:** $s_i = 0$ and y_i being indeterminate.

If $\mathbf{x}$ is infeasible for (3), i.e., $\mathcal{I}(\mathbf{x}) < \delta$, condition **a1** ensures that (10b) is violated, hence $\mathbf{x}$ is also infeasible for (10). Conversely, if $\mathbf{x}$ is infeasible for (10), i.e., $\sum_i y_i < \delta$, then $\mathcal{I}(\mathbf{x}) < \delta$, hence $\mathbf{x}$ is also infeasible for (3).

If $\mathbf{x}$ is feasible for (3), we can always set $y_i = 1$ and $s_i = 0$ for all inliers to satisfy (10b), ensuring the feasibility of $\mathbf{x}$ to (10). Conversely, if $\mathbf{x}$ is feasible for (10), by **a1** there are at least δ inliers, thus $\mathbf{x}$ is also feasible to (3). □

From the computational standpoint, (10) is no easier to solve than (3). However, by constructing a cost function from the bilinear constraints (10d), we

arrive at the following continuous optimization problem

$$\underset{\mathbf{x}\in D,\ \mathbf{s}\in\mathbb{R}^N,\ \mathbf{y}\in\mathbb{R}^N}{\text{minimize}} \quad \sum_i y_i s_i \tag{11a}$$

$$\text{subject to} \quad \sum_i y_i \geq \delta, \tag{11b}$$

$$y_i \in [0,1], \ \ \forall i, \tag{11c}$$

$$s_i - r_i'(\mathbf{x}) \geq 0, \ \ \forall i, \tag{11d}$$

$$s_i \geq 0, \ \ \forall i, \tag{11e}$$

where $\mathbf{s} = [s_1, \ldots, s_N]^T$ and $\mathbf{y} = [y_1, \ldots, y_N]^T$. The following lemma establishes the equivalence between (11) and (3).

Lemma 2. *If the globally optimal value of* (11) *is zero, then there exists* $\mathbf{x}$ *that satisfies the update problem* (3)*.*

Proof. Due to (11c) and (11e), the objective value of (11) is lower bounded by zero. Let $(\mathbf{x}^*, \mathbf{s}^*, \mathbf{y}^*)$ be a global minimizer of (11). If $\sum_i y_i^* s_i^* = 0$, then $\mathbf{x}^*$ satisfies all the constraints in (10), thus $\mathbf{x}^*$ is feasible to (3). □

3.2 Biconvex optimization algorithm

Although all the constraints in (11) are convex (including $\mathbf{x} \in D$), the objective function is not convex. Nonetheless, the primary value of formulation (11) is to enable the usage of convex solvers to approximately solve the update problem. Note also that (11) does not require any smoothing parameters.

To this end, observe that (11) is in fact an instance of *biconvex programming* [1]. If we fix $\mathbf{x}$ and $\mathbf{s}$, (11) reduces to the linear program (LP)

$$\underset{\mathbf{y}\in\mathbb{R}^N}{\text{minimize}} \quad \sum_i y_i s_i \tag{12a}$$

$$\text{subject to} \quad \sum_i y_i \geq \delta, \tag{12b}$$

$$y_i \in [0,1], \ \ \forall i, \tag{12c}$$

which can be solved in close form.[6] On the other hand, if we fix $\mathbf{y}$, (11) reduces to the second order cone program (SOCP)

$$\underset{\mathbf{x}\in D,\mathbf{s}\in\mathbb{R}^N}{\text{minimize}} \quad \sum_i y_i s_i \tag{13a}$$

$$\text{subject to} \quad s_i - r_i'(\mathbf{x}) \geq 0, \ \ \forall i, \tag{13b}$$

$$s_i \geq 0, \ \ \forall i. \tag{13c}$$

[6] Set $y_i = 1$ if s_i is one of the δ-smallest slacks, and $y_i = 0$ otherwise.

Algorithm 1 Biconvex optimization (BCO) for the continuous problem (11).

Require: Initial solution $\tilde{\mathbf{x}}$, target consensus δ.
1: Initialize $\hat{\mathbf{x}} \leftarrow \tilde{\mathbf{x}}$, set $\hat{\mathbf{s}}$ using (14).
2: **while** not converged **do**
3: $\quad \hat{\mathbf{y}} \leftarrow$ solve LP (12).
4: $\quad (\hat{\mathbf{x}}, \hat{\mathbf{s}}) \leftarrow$ solve SOCP (13).
5: **end while**
6: **return** $\hat{\mathbf{x}}$, $\hat{\mathbf{s}}$ and $\hat{\mathbf{y}}$.

Note that s_i does not have influence if the corresponding $y_i = 0$; these slack variables can be removed from the problem to speed up optimization.[7]

The proposed algorithm (called *Biconvex Optimization* or *BCO*) is simple: we initialize $\mathbf{x}$ as the starting $\tilde{\mathbf{x}}$ from (3), and set the slacks as

$$s_i = \max\{0, r_i'(\tilde{\mathbf{x}})\}, \quad \forall i. \tag{14}$$

Then, we alternate between solving the LP and SOCP until convergence. Since (11) is lower-bounded by zero, and each invocation of the LP and SOCP are guaranteed to reduce the cost, BCO will always converge to a *local optimum* $(\hat{\mathbf{x}}, \hat{\mathbf{s}}, \hat{\mathbf{y}})$.

In respect to solving the update problem (3), if the local optimum $(\hat{\mathbf{x}}, \hat{\mathbf{s}}, \hat{\mathbf{y}})$ turns out to be the global optimum (i.e., $\sum_i \hat{y}_i \hat{s}_i = 0$), then $\hat{\mathbf{x}}$ is a solution to (3), i.e., $\mathcal{I}(\hat{\mathbf{x}}) \geq \delta$. Else, $\hat{\mathbf{x}}$ might still represent an improved solution over $\tilde{\mathbf{x}}$. Compared to randomized search, our method is by design more capable of improving $\tilde{\mathbf{x}}$. This is because optimizing (11) naturally reduces the residual of outliers that "should be" an inlier, i.e., with $y_i = 1$, which may still lead to a local refinement, i.e., $\mathcal{I}(\hat{\mathbf{x}}) > \delta_l = \mathcal{I}(\tilde{\mathbf{x}})$, regardless of whether problem (3) is feasible or not. In the next section, we will construct an effective deterministic consensus maximization technique based on Algorithm 1.

4 Main algorithm—deterministic consensus maximization

Given an initial solution $\mathbf{x}^{(0)}$ to (1), e.g., obtained using least squares or a random sampling heuristic, we wish to update $\mathbf{x}^{(0)}$ to a better solution. The main structure of our proposed algorithm is simple: we conduct bisection over the consensus value to search for a better solution; see Algorithm 2.

A lower and upper bound δ_l and δ_h for the consensus, which are initialized respectively to $\mathcal{I}(\mathbf{x}^{(0)})$ and N, are maintained and progressively tightened. Let $\tilde{\mathbf{x}}$ be the current best solution (initialized to $\mathbf{x}^{(0)}$); then, the midpoint $\delta = \lfloor 0.5(\delta_l + \delta_h) \rfloor$ is obtained and the update problem via the continuous biconvex formulation (11) is solved using Algorithm 1. If the solution $\hat{\mathbf{x}}$ for (11) has a higher quality than the incumbent, $\tilde{\mathbf{x}}$ is revised to become $\hat{\mathbf{x}}$ and δ_l is increased to $\mathcal{I}(\hat{\mathbf{x}})$. And if $\mathcal{I}(\hat{\mathbf{x}}) < \delta$, δ_h is decreased to δ. Algorithm 2 ends when $\delta_h = \delta_l + 1$.

[7] Given the optimal $\hat{\mathbf{x}}$ for (13), the values of the slack variables that did not participate in the problem can be obtained as $s_i = \max\{0, r_i'(\hat{\mathbf{x}})\}$.

Algorithm 2 Bisection (non-global) for deterministic consensus maximization.

Require: Initial solution $\mathbf{x}^{(0)}$ for (1) obtained using least squares or random sampling.
1: $\tilde{\mathbf{x}} \leftarrow \mathbf{x}^{(0)}$, $\delta_h \leftarrow N$, $\delta_l \leftarrow \mathcal{I}(\mathbf{x}^{(0)})$.
2: **while** $\delta_h > \delta_l + 1$ **do**
3: $\quad \delta \leftarrow \lfloor 0.5(\delta_l + \delta_h) \rfloor$.
4: $\quad (\hat{\mathbf{x}}, \hat{\mathbf{s}}, \hat{\mathbf{y}}) \leftarrow \text{BCO}(\tilde{\mathbf{x}}, \delta)$ (see Algorithm 1).
5: $\quad$ **if** $\mathcal{I}(\hat{\mathbf{x}}) > \mathcal{I}(\tilde{\mathbf{x}})$ **then**
6: $\quad\quad \tilde{\mathbf{x}} \leftarrow \hat{\mathbf{x}}$, $\delta_l \leftarrow \mathcal{I}(\hat{\mathbf{x}})$.
7: $\quad$ **end if**
8: $\quad$ **if** $\mathcal{I}(\hat{\mathbf{x}}) < \delta$ **then**
9: $\quad\quad \delta_h \leftarrow \delta$.
10: $\quad$ **end if**
11: **end while**
12: **return** $\tilde{\mathbf{x}}$, δ_l.

Since the "feasibility test" in Algorithm 2 (Step 4) is solved via a non-convex subroutine, the bisection technique does not guarantee finding the global solution, i.e., the quality of the final solution may underestimate the maximum achievable quality. However, our technique is fundamentally advantageous compared to previous methods [9, 17, 22] since it is not subject to the vagaries of randomization or require tuning of hyperparameters. Empirical results in the next section will demonstrate the effectiveness of the proposed algorithm.

5 Results

We call the proposed algorithm *IBCO* (for *iterative biconvex optimization*). We compared IBCO against the following random sampling methods:

- RANSAC (RS) [11] (baseline): the confidence ρ was set to 0.99 for computing the termination threshold.
- PROSAC (PS) [8] and Guided MLESAC (GMS) [26] (RS variants with guided sampling): only tested for fundamental matrix and homography estimation since inlier priors like matching scores for correspondences were needed.
- LO-RANSAC (LRS) [9]: subset size in inner sampling was set to half of the current consensus size, and the max number of inner iterations was set to 10.
- Fixing LO-RANSAC (FLRS) [18]: subset size in inner sampling was set to 7× minimal subset size, and the max number of inner iterations was set to 50.
- USAC [23]: a modern technique that combines ideas from PS and LRS.[8] USAC was evaluated only on fundamental matrix and homography estimation since the available code only implements these models.

Except USAC which was implemented in C++, the other sampling methods were based on MATLAB [16]. Also, least squares was executed on the final consensus set to refine the results of all the random sampling methods.

[8] Code from `https://http://www.cs.unc.edu/~rraguram/usac/USAC-1.0.zip`.

Convex subproblem	LP	SOCP
Solvers used	Gurobi	Sedumi
Methods using the solver	EP, SS	IBCO

Table 1: Convex solvers used in deterministic methods.

In addition to the random sampling methods, we also compared IBCO against the following deterministic consensus maximization algorithms:

- Exact Penalty (EP) method [17]: The method[9] was retuned for best performance on our data: we set the penalty parameter α to 1.5 for fundamental matrix estimation and 0.5 for all other problems. The annealing rate κ for the penalty parameter was set to 5 for linear regression and 2D homography estimation and 1.5 for triangulation and fundamental matrix estimation.
- Smooth Surrogate (SS) method [22]: Using our own implementation. The smoothing parameter γ was set to 0.01 as suggested in [22].

For the deterministic methods, Table 1 lists the convex solvers used for their respective subproblems. Further, results for these methods with both FLRS and random initialization ($\mathbf{x}^{(0)}$ was generated randomly) were provided, in order to show separately the performance with good (FLRS) and bad (random) initialization. We also tested least squares initialization, but under high outlier rates, its effectiveness was no better than random initialization. All experiments were executed on a laptop with Intel Core 2.60GHz i7 CPU and 16GB RAM.

5.1 Robust linear regression on synthetic data

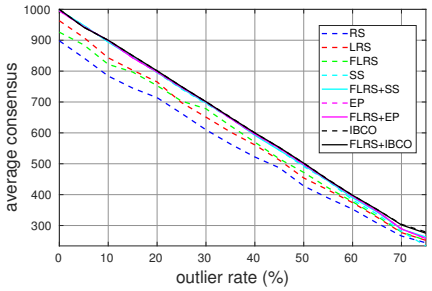
Data of size $N = 1000$ for 8-dimensional linear regression (i.e., $\mathbf{x} \in \mathbb{R}^8$) were synthetically generated. In linear regression, the residual takes the form

$$r_i(\mathbf{x}) = \|\mathbf{a}_i^T\mathbf{x} - b_i\|_2, \tag{15}$$

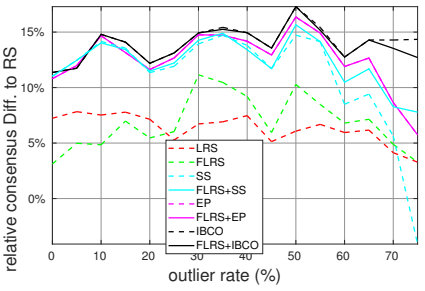
which is a special case of (4) (set $p_i(\mathbf{x}) = 1$), and each datum is represented by $\{\mathbf{a}_i \in \mathbb{R}^8,\, b_i \in \mathbb{R}\}$. First, the independent measurements $\{\mathbf{a}_i\}_{i=1}^N$ and parameter vector $\mathbf{x}$ were randomly sampled. The dependent measurements were computed as $b_i = \mathbf{a}_i^T\mathbf{x}$ and added with noise uniformly distributed between $[-0.3, 0.3]$. A subset of $\eta\%$ of the dependent measurements were then randomly selected and added with Gaussian noise of $\sigma = 1.5$ to create outliers. To guarantee the outlier rate, each outlier is regenerated until the noise is not within [-0.3,0.3]. The inlier threshold ϵ for (1) was set to 0.3.

Fig. 2 shows the optimized consensus, runtime and model accuracy of the methods for $\eta \in \{0, 5, ..., 70, 75\}$, averaged over 10 runs for each data instance. Note that the actual outlier rate was sometimes slightly lower than expected since the largest consensus set included some outliers with low noise value. For $\eta = 75$ the actual outlier rate was around 72% (see Fig. 2(a)). To prevent inaccurate analysis caused by this phenomenon, results for $\eta > 75$ were not provided.

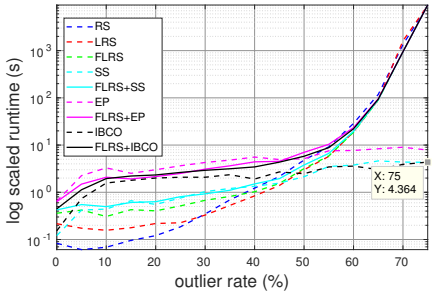
[9] Code from `https://cs.adelaide.edu.au/~huu/`.



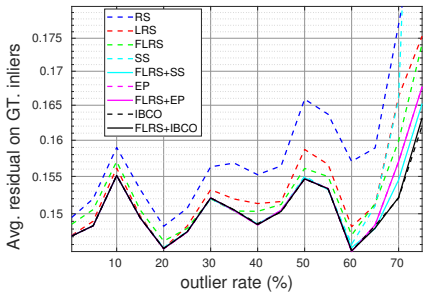
(a) Average optimized consensus.



(b) Relative difference of consensus to RS.



(c) Average runtime (seconds, in log scale).



(d) Average residuals on ground truth inliers for models fitted on the consensus set by least squares.

Fig. 2: Robust linear regression results with varied η (approx. outlier rate).

Fig. 2(b) demonstrates for each method the relative consensus difference to RS. It is evident that both IBCO variants outperformed other methods in general. Unlike other methods, whose improvement to RS was low at high outlier rates, both IBCO variants were consistently better than RS by more than 11%. Though IBCO was only marginally better than EP for outlier rates lower than 65%, Fig. 2(a) shows that for most of the data instances, both IBCO variants found consensus very close or exactly equal to the maximum achievable. The cost of IBCO was fairly practical (less than 5 seconds for all data instances, see the data tip in Fig. 2(c)). Also the runtime of the random sampling methods (RS, LRS, FLRS) increased exponentially with η. Hence, at high η, the major cost of FLRS+EP, FLRS+SS and FLRS+IBCO came from FLRS.

To demonstrate the significance of having higher consensus, we further performed least squares fitting on the consensus set of each method. Given a least squares fitted model $\mathbf{x}_{LS}$, define the average residual on ground truth inliers (the data assigned with less than 0.3 noise level) as:

$$e(\mathbf{x}_{LS}) = \frac{\sum_{i^* \in \mathcal{I}^*} r_{i^*}(\mathbf{x}_{LS})}{|\mathcal{I}^*|}, \tag{16}$$

where $\mathcal{I}^*$ was the set of all ground truth inliers. Fig. 2(d) shows $e(\mathbf{x}_{LS})$ for all methods on all data instances. Generally, higher consensus led to a lower average residual, suggesting a more accurate model.

5.2 Homography estimation

Five image pairs from the NYC Library dataset [29] were used for 2D homography estimation. On each image pair, SIFT correspondences were produced by the VLFeat toolbox [28] and used as inputs. Fig. 3 depicts examples of inputs, as well as consensus sets from FLRS and FLRS+IBCO. The transfer error in one image [12, Sec. 4.2.2] was used as the distance measurement. The inlier threshold ϵ was set to 4 pixels. The 4-Point algorithm [12, Sec. 4.7.1] was used in all random sampling approaches for model fitting on minimal samples.

Fig. 4, shows the quantitative results, averaged over 50 runs. Though marginally costlier than SS and random approaches, both IBCO variants found considerably larger consensus sets than other methods for all data. Meanwhile, different from the linear regression case, EP no longer had simiar result quality to IBCO. Also note that for challenging problems, e.g., *Ceiling1* and *Sign*, the two IBCO variants were the only methods that returned much higher consensus than RS.

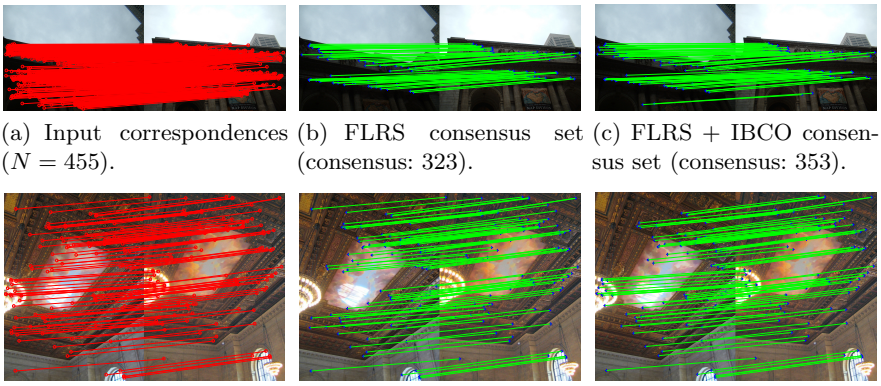
5.3 Triangulation

Five feature tracks from the NotreDame dataset [24] were selected for triangulation, i.e., estimating the 3D coordinates. The input from each feature track contained a set of camera matrices and the corresponding 2D feature coordinates. The re-projection error was used as the distance measurement [15] and the inlier threshold ϵ was set to 1 pixel. The size of minimal samples was 2 (views) for all RANSAC variants. The results are demonstrated in Fig. 5. For triangulation, the quality of the initial solution largely affected the performance of EP, SS and IBCO. Initialized with FLRS, IBCO managed to find much larger consensus sets than all other methods.

5.4 Effectiveness of refinement

Though all deterministic methods were provided with reliable initial FLRS solutions, IBCO was the only one that effectively refined all FLRS results. EP and SS sometimes even converged to worse than initial solutions. To illustrate these effects, Fig. 6 shows the solution quality during the iterations of the three deterministic methods (initialized by FLRS) on *Ceiling1* for homography estimation and *Point 16* for triangulation. In contrast to EP and SS which progressively made the initial solution worse, IBCO steadily improved the initial solution.

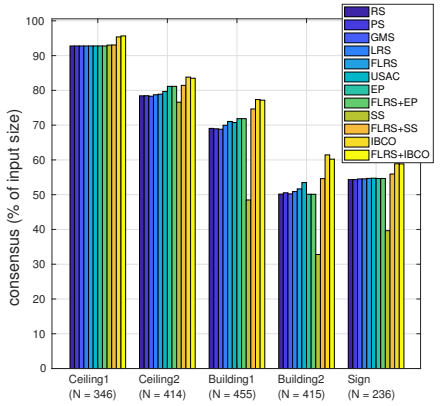
It may be possible to rectify the behaviour of EP and SS by choosing more appropriate smoothing parameters and/or their annealing rates. However, the need for data-dependent tuning makes EP and SS less attractive than IBCO.



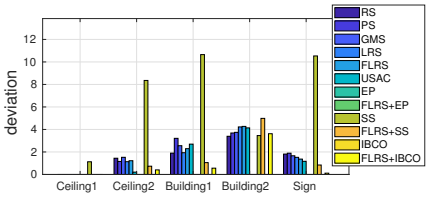
(a) Input correspondences ($N = 455$). (b) FLRS consensus set (consensus: 323). (c) FLRS + IBCO consensus set (consensus: 353).

(d) Input correspondences ($N = 346$). (e) FLRS consensus set (consensus: 321). (f) FLRS + IBCO consensus set (consensus: 331).

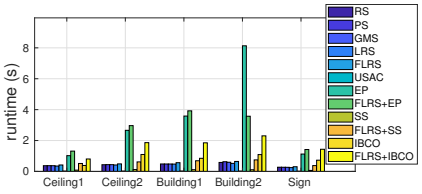
Fig. 3: Data and results of robust homography estimation for *Building1* (top) and *Ceiling1* (bottom). Consensus sets were downsampled for visual clarity.



(a) Average optimized consensus (as % of input size N). N is provided in the brackets.



(b) Standard deviation of optimized consensus over 50 runs.



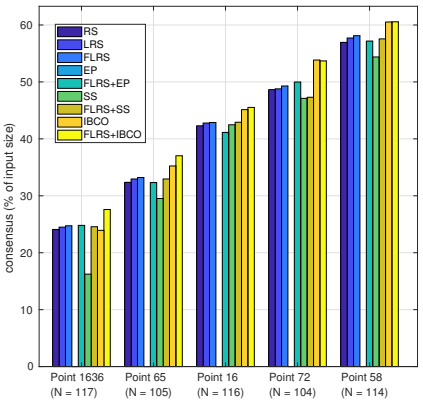
(c) Runtime in seconds.

Fig. 4: Robust homography estimation results.

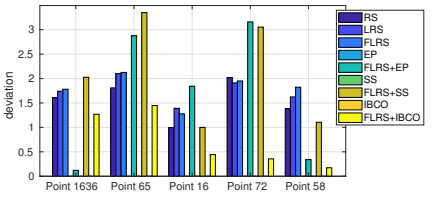
5.5 Fundamental matrix estimation

Image pairs from the two-view geometry corpus of CMP[10] were used for fundamental matrix estimation. As in homography estimation, SIFT correspondences were used as the input data. Since the Sampson error [12, Sec. 11.4.3] and the reprojection error [12, Sec. 11.4.1] for fundamental matrix estimation are not linear or quasiconvex, the deterministic algorithms (EP, SS, IBCO) cannot be

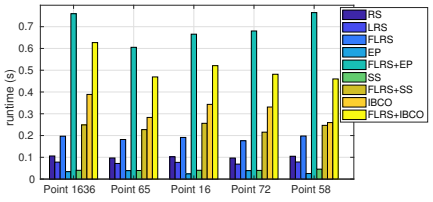
[10] `http://cmp.felk.cvut.cz/data/geometry2view/`



(a) Average optimized consensus (as % of input size N). N is provided in the brackets.

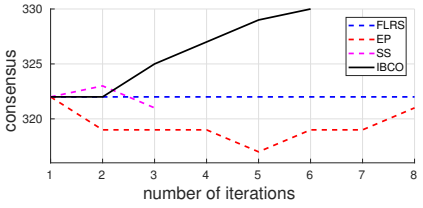


(b) Standard deviation of optimized consensus over 50 runs.

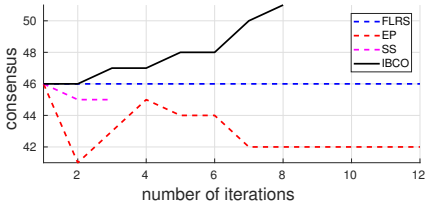


(c) Runtime in seconds.

Fig. 5: Robust triangulation results.



(a) *Ceiling1* in homography estimation.



(b) *Point 16* in triangulation.

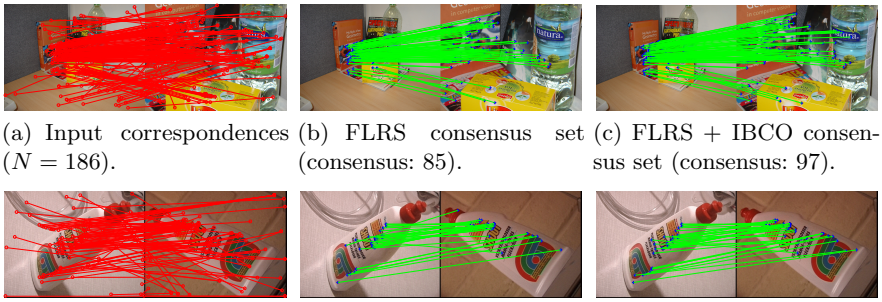
Fig. 6: Consensus size in each iteration, given FLRS results as the initialization. Observe that EP and SS converged to worse off solutions.

directly applied. Thus, we linearize the epipolar constraint and use the algebraic error [12, Sec. 11.3] as the residual. The inlier threshold ϵ was set to 0.006 for all data.

Further, a valid fundamental matrix satisfies the rank-2 constraint [12, Sec. 11.1.1], which is non-convex. For EP, SS, IBCO, we impose the rank-2 constraint using SVD after each parameter vector updates (for IBCO, after each BCO run).

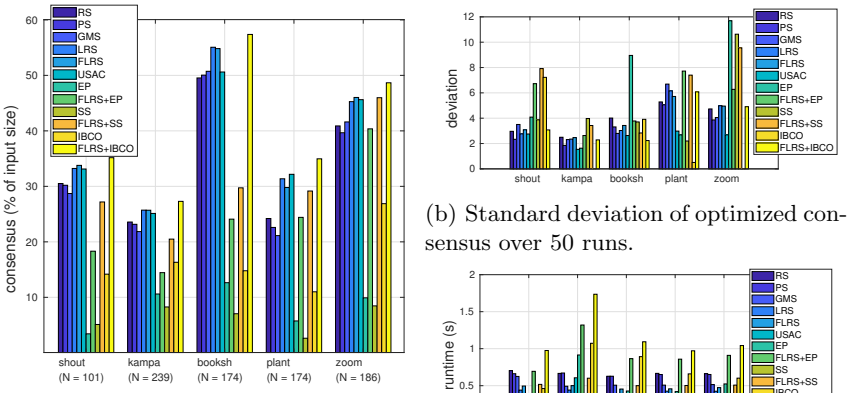
Fig. 7 depicts sample image pairs and generated SIFT correspondences, as well as consensus sets from FLRS and FLRS+IBCO. The seven-point method [12, Sec. 11.1.2] was used in USAC and the normalized 8-point algorithm [12, Sec. 11.2] was used in all other RANSAC variants.

As shown in Fig. 8(a), unlike EP and SS who failed to refine the initial FLRS results for all the tested data, IBCO was still effective even though the problem contains non-convex constraints.



(a) Input correspondences ($N = 186$).
(b) FLRS consensus set (consensus: 85).
(c) FLRS + IBCO consensus set (consensus: 97).
(d) Input correspondences ($N = 101$).
(e) FLRS consensus set (consensus: 32).
(f) FLRS + IBCO consensus set (consensus: 36).

Fig. 7: Data and results of fundamental matrix estimation for *zoom* (top) and *shout* (bottom).



(a) Average optimized consensus (as % of input size N). N is provided in the brackets.
(b) Standard deviation of optimized consensus over 50 runs.
(c) Runtime in seconds.

Fig. 8: Robust fundamental matrix estimation results.

6 Conclusions

We proposed a novel deterministic algorithm for consensus maximization with non-linear residuals. The basis of our method lies in reformulating the decision version of consensus maximization into an instance of biconvex programming, which enables the use of bisection for efficient guided search. Compared to other deterministic methods, our method does not relax the objective of consensus maximization problem and is free from the tuning of smoothing parameters, which makes it much more effective at refining the initial solution. Experiments show that our method is able to greatly improve upon initial results from widely used random sampling heuristics.

Acknowledgements This work was supported by the ARC grant DP160103490.

References

1. Biconvex optimization. https://en.wikipedia.org/wiki/Biconvex_optimization
2. Boyd, S., Vandenberghe, L.: Convex optimization. Cambridge university press (2004)
3. Campbell, D., Petersson, L., Kneip, L., Li, H.: Globally-optimal inlier set maximisation for simultaneous camera pose and feature correspondence. arXiv preprint arXiv:1709.09384 (2017)
4. Chin, T.J., Cai, Z., Neumann, F.: Robust fitting in computer vision: Easy or hard? arXiv preprint arXiv:1802.06464 (2018)
5. Chin, T.J., Heng Kee, Y., Eriksson, A., Neumann, F.: Guaranteed outlier removal with mixed integer linear programs. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 5858–5866 (2016)
6. Chin, T.J., Purkait, P., Eriksson, A., Suter, D.: Efficient globally optimal consensus maximisation with tree search. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2413–2421 (2015)
7. Choi, S., Kim, T., Yu, W.: Performance evaluation of RANSAC family. In: British Machine Vision Conference (BMVC) (2009)
8. Chum, O., Matas, J.: Matching with prosac-progressive sample consensus. In: 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05). vol. 1, pp. 220–226. IEEE (2005)
9. Chum, O., Matas, J., Kittler, J.: Locally optimized ransac. In: DAGM. Springer (2003)
10. Enqvist, O., Ask, E., Kahl, F., Åström, K.: Tractable algorithms for robust model estimation. International Journal of Computer Vision **112**(1), 115–129 (2015)
11. Fischler, M.A., Bolles, R.C.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Communications of the ACM **24**(6), 381–395 (1981)
12. Hartley, R., Zisserman, A.: Multiple view geometry in computer vision. Cambridge university press (2003)
13. Hu, J., Mitchell, J.E., Pang, J.S., Yu, B.: On linear programs with linear complementarity constraints. Journal of Global Optimization **53**(1), 29–51 (2012)
14. Kahl, F., Hartley, R.: Multiple-view geometry under the l_∞-norm. IEEE Transactions on Pattern Analysis and Machine Intelligence **30**(9), 1603–1617 (2008)
15. Ke, Q., Kanade, T.: Quasiconvex optimization for robust geometric reconstruction. IEEE Transactions on Pattern Analysis and Machine Intelligence **29**(10) (2007)
16. Kovesi, P.D.: MATLAB and Octave functions for computer vision and image processing, available from: <http://www.peterkovesi.com/matlabfns/>
17. Le, H., Chin, T.J., Suter, D.: An exact penalty method for locally convergent maximum consensus. In: Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on. IEEE (2017)
18. Lebeda, K., Matas, J., Chum, O.: Fixing the locally optimized ransac–full experimental evaluation. In: British Machine Vision Conference. pp. 1–11. Citeseer (2012)
19. Li, H.: Consensus set maximization with guaranteed global optimality for robust geometry estimation. In: Computer Vision, 2009 IEEE 12th International Conference on. pp. 1074–1080. IEEE (2009)
20. Meer, P.: Robust techniques for computer vision. Emerging topics in computer vision pp. 107–190 (2004)

21. Olsson, C., Enqvist, O., Kahl, F.: A polynomial-time bound for matching and registration with outliers. In: Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on. pp. 1–8. IEEE (2008)
22. Purkait, P., Zach, C., Eriksson, A.: Maximum consensus parameter estimation by reweighted ℓ_1 methods. In: International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition. pp. 312–327. Springer (2017)
23. Raguram, R., Chum, O., Pollefeys, M., Matas, J., Frahm, J.M.: Usac: a universal framework for random sample consensus. IEEE transactions on pattern analysis and machine intelligence **35**(8), 2022–2038 (2013)
24. Snavely, N., Seitz, S.M., Szeliski, R.: Photo tourism: exploring photo collections in 3d. In: ACM transactions on graphics (TOG). vol. 25, pp. 835–846. ACM (2006)
25. Speciale, P., Paudel, D.P., Oswald, M.R., Kroeger, T., Gool, L.V., Pollefeys, M.: Consensus maximization with linear matrix inequality constraints. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 5048–5056. IEEE (2017)
26. Tordoff, B.J., Murray, D.W.: Guided-mlesac: Faster image transform estimation by using matching priors. IEEE transactions on pattern analysis and machine intelligence **27**(10), 1523–1535 (2005)
27. Tran, Q.H., Chin, T.J., Chojnacki, W., Suter, D.: Sampling minimal subsets with large spans for robust estimation. International journal of computer vision **106**(1), 93–112 (2014)
28. Vedaldi, A., Fulkerson, B.: VLFeat: An open and portable library of computer vision algorithms. `http://www.vlfeat.org/` (2008)
29. Wilson, K., Snavely, N.: Robust global translations with 1dsfm. In: Proceedings of the European Conference on Computer Vision (ECCV) (2014)
30. Zheng, Y., Sugimoto, S., Okutomi, M.: Deterministically maximizing feasible subsystem for robust model fitting with unit norm constraint. In: Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on. pp. 1825–1832. IEEE (2011)

Chapter 5

Consensus Maximization Tree Search Revisited

The work contained in this chapter has been published as the following papers

Zhipeng Cai, Tat-Jun Chin, Vladlen Koltun, Consensus Maximization Tree Search Revisited, International Conference on Computer Vision (ICCV) 2019.

Statement of Authorship

Title of Paper	Consensus Maximization Tree Search Revisited
Publication Status	☑ Published ☐ Accepted for Publication ☐ Submitted for Publication ☐ Unpublished and Unsubmitted work written in manuscript style
Publication Details	Cai, Zhipeng, Tat-Jun Chin, and Vladlen Koltun. "Consensus Maximization Tree Search Revisited." Proceedings of the IEEE International Conference on Computer Vision. 2019.

Principal Author

Name of Principal Author (Candidate)	Zhipeng Cai		
Contribution to the Paper	Proposing the main idea. Conducting experiments. Paper writing.		
Overall percentage (%)	60%		
Certification:	This paper reports on original research I conducted during the period of my Higher Degree by Research candidature and is not subject to any obligations or contractual agreements with a third party that would constrain its inclusion in this thesis. I am the primary author of this paper.		
Signature		Date	16/2/2020

Co-Author Contributions

By signing the Statement of Authorship, each author certifies that:

i. the candidate's stated contribution to the publication is accurate (as detailed above);
ii. permission is granted for the candidate in include the publication in the thesis; and
iii. the sum of all co-author contributions is equal to 100% less the candidate's stated contribution.

Name of Co-Author	Tat-Jun Chin		
Contribution to the Paper	Providing suggestions about the method and experiments. Modifications of the paper draft.		
Signature		Date	15/2/2020

Name of Co-Author	Vladlen Koltun		
Contribution to the Paper	Providing major discussions and suggestions. Proofreading of the paper draft.		
Signature		Date	18/2/2020

Consensus Maximization Tree Search Revisited

Zhipeng Cai
The University of Adelaide

Tat-Jun Chin
The University of Adelaide

Vladlen Koltun
Intel Labs

Abstract

Consensus maximization is widely used for robust fitting in computer vision. However, solving it exactly, i.e., finding the globally optimal solution, is intractable. A tree search, which has been shown to be fixed-parameter tractable, is one of the most efficient exact methods, though it is still limited to small inputs. We make two key contributions towards improving A* tree search. First, we show that the consensus maximization tree structure used previously actually contains paths that connect nodes at both adjacent and non-adjacent levels. Crucially, paths connecting non-adjacent levels are redundant for tree search, but they were not avoided previously. We propose a new acceleration strategy that avoids such redundant paths. In the second contribution, we show that the existing branch pruning technique also deteriorates quickly with the problem dimension. We then propose a new branch pruning technique that is less dimension-sensitive to address this issue. Experiments show that both new techniques can significantly accelerate A* tree search, making it reasonably efficient on inputs that were previously out of reach. Demo code is available at* `https://github.com/ZhipengCai/MaxConTreeSearch`.

1. Introduction

The prevalence of outliers makes robust model fitting crucial in many computer vision applications. One of the most popular robust fitting criteria is *consensus maximization*, whereby, given outlier-contaminated data $\mathcal{S} = \{\mathbf{s}_i\}_{i=1}^N$, we seek the model $\boldsymbol{\theta} \in \mathbb{R}^d$ that is consistent with the largest subset of the data. Formally, we solve

$$\underset{\boldsymbol{\theta}}{\text{maximize}} \quad c(\boldsymbol{\theta}|\mathcal{S}) = \sum_{i=1}^{N} \mathbb{I}\{r(\boldsymbol{\theta}|\mathbf{s}_i) \leq \epsilon\}, \qquad (1)$$

where $c(\boldsymbol{\theta}|\mathcal{S})$ is called the *consensus* of $\boldsymbol{\theta}$. The 0/1 valued indicator function $\mathbb{I}\{\cdot\}$ returns 1 only when $\mathbf{s}_i$ is consistent with $\boldsymbol{\theta}$, which happens when the residual $r(\boldsymbol{\theta}|\mathbf{s}_i) \leq \epsilon$. The form of $r(\boldsymbol{\theta}|\mathbf{s}_i)$ will be defined later in Sec. 2. Constant ϵ is the predefined inlier threshold, and d is called the "problem dimension". Given the optimal solution $\boldsymbol{\theta}^*$ of (1), $\mathbf{s}_i$ is an inlier if $r(\boldsymbol{\theta}^*|\mathbf{s}_i) \leq \epsilon$ and an outlier otherwise.

Consensus maximization is NP-hard [4], hence, sub-optimal but efficient methods are generally more practical. Arguably the most prevalent methods of this type are RANSAC [11] and its variants [8, 26, 7, 24], which iteratively fit models on randomly sampled (minimal) data subsets and return the model with the highest consensus. However, their inherent randomness makes these methods often distant from optimal and sometimes unstable. To address this problem, deterministic optimization techniques [23, 14, 2] have been proposed, which, with good initializations, usually outperform RANSAC variants. Nonetheless, a good initial solution is not always easy to find. Hence, these methods may still return unsatisfactory results.

The weaknesses of sub-optimal methods motivate researchers to investigate globally optimal methods; however, so far they are effective on only small input sizes (small d, N and/or number of outliers o). One of the most efficient exact methods is tree search [15, 5, 6] (others surveyed later in Sec. 1.1), which fits (1) into the framework of the LP-type methods [25, 18]. By using heuristics to guide the tree search and conduct branch pruning, A* tree search [5, 6] has been demonstrated to be much faster than Breadth-First Search (BFS) and other types of globally optimal algorithms. In fact, tree search is provably fixed-parameter tractable (FPT) [4]. Nevertheless, as demonstrated in the experiment of [6] and later ours, A* tree search can be highly inefficient for challenging data with moderate d (≥ 6) and o (≥ 10).

Our contributions. In this work, we analyze reasons behind the inefficiency of A* tree search and develop improvements to the algorithm. Specifically:

- We demonstrate that the previous tree search algorithm does not avoid all redundant paths, namely, paths that connect nodes from non-adjacent levels. Based on this observation, a new acceleration strategy is proposed, which can avoid such non-adjacent (and redundant) paths.
- We show that the branch pruning technique in [6] is not always effective and may sometimes slow down the tree search due to its sensitivity to d. To address this problem, we propose a branch pruning technique that is less dimension-sensitive and hence much more effective.

Experiments demonstrate the significant acceleration achievable using our new techniques (3 orders of magnitude

faster on challenging data). Our work represents significant progress towards making globally optimal consensus maximization practical on real data.

1.1. Related Work

Besides tree search, other types of globally optimal methods include branch-and-bound (BnB) [16, 28, 22], whose exhaustive search is done by testing all possible $\boldsymbol{\theta}$. However, the time complexity of BnB is exponential in the size of the parameter space, which is often large. Moreover, the bounding function of BnB is problem-dependent and not always trivial to construct. Another type of methods [20, 9] enumerate and fit models on all possible bases, where each basis is a data subset of size p, where $p \ll N$ and p is usually slightly larger than d, e.g., $p = d + 1$. The number of all possible bases is $\binom{N}{p}$, which scales poorly with N and d. Besides differences in actual runtime, what distinguishes tree search from the other two types of methods is that tree search is FPT [4]: its worst case runtime is exponential in d and o, but polynomial in N.

2. Consensus maximization tree search

We first review several concepts that are relevant to consensus maximization tree search.

2.1. Application range

Tree search requires the residual $r(\boldsymbol{\theta}|\mathbf{s}_i)$ to be *pseudo-convex* [6]. A simple example is the linear regression residual

$$r(\boldsymbol{\theta}|\mathbf{s}_i) = |\mathbf{a}_i^T\boldsymbol{\theta} - b_i|, \tag{2}$$

where each datum $\mathbf{s}_i = \{\mathbf{a}_i, b_i\}$, $\mathbf{a}_i \in \mathbb{R}^d$ and $b_i \in \mathbb{R}$. Another example is the residual used in common multiview geometry problems [21, 2], which are of the form

$$r(\boldsymbol{\theta}|\mathbf{s}_i) = \frac{\left\|\mathbf{A}_i^T\boldsymbol{\theta} - \mathbf{b}_i\right\|_p}{\mathbf{c}_i^T\boldsymbol{\theta} - d_i}, \tag{3}$$

where each datum $\mathbf{s}_i = \{\mathbf{A}_i, \mathbf{b}_i, \mathbf{c}_i, d_i\}$, $\mathbf{A}_i \in \mathbb{R}^{d\times m}, \mathbf{b}_i \in \mathbb{R}^m$, $\mathbf{c}_i \in \mathbb{R}^d$ and $d_i \in \mathbb{R}$. Usually, p is 1, 2 or ∞.

2.2. LP-type problem

The tree search algorithm for (1) is constructed by solving a series of minimax problems, which are of the form

$$\underset{\boldsymbol{\theta}}{\text{minimize}} \ \max_{i\in\mathcal{S}^1} \ r(\boldsymbol{\theta}|\mathbf{s}_i). \tag{4}$$

Problem (4) minimizes the maximum residual for all data in $\mathcal{S}^1$, which is an arbitrary subset of $\mathcal{S}$. For convenience, we define $f(\mathcal{S}^1)$ as the minimum objective value of (4) computed on data $\mathcal{S}^1$, and $\boldsymbol{\theta}(\mathcal{S}^1)$ as the (exact) minimizer.

Throughout the paper, we will assume that $r(\cdot)$ is pseudo-convex and $\mathcal{S}$ is non-degenerate (otherwise infinitestimal perturbations can be applied to remove degeneracy [18, 6]). Under this assumption, problem (4) has a unique optimal solution and can be solved efficiently with standard solvers [10]. Furthermore, (4) is provably an LP-type problem [25, 1, 10], which is a generalization of the linear programming (LP) problem. An LP-type problem has the following properties:

Property 1 (Monotonicity). *For every two sets $\mathcal{S}^1 \subseteq \mathcal{S}^2 \subseteq \mathcal{S}$, $f(\mathcal{S}^1) \leq f(\mathcal{S}^2) \leq f(\mathcal{S})$.*

Property 2 (Locality). *For every two sets $\mathcal{S}^1 \subseteq \mathcal{S}^2 \subseteq \mathcal{S}$ and every $\mathbf{s}_i \in \mathcal{S}$, $f(\mathcal{S}^1) = f(\mathcal{S}^2) = f(\mathcal{S}^2 \cup \{\mathbf{s}_i\}) \Rightarrow f(\mathcal{S}^1) = f(\mathcal{S}^1 \cup \{\mathbf{s}_i\})$.*

With the above properties, the concept of *basis*, which is essential for tree search, can be defined.

Definition 1 (Basis). *A basis $\mathcal{B}$ in $\mathcal{S}$ is a subset of $\mathcal{S}$ such that for every $\mathcal{B}' \subset \mathcal{B}$, $f(\mathcal{B}') < f(\mathcal{B})$.*

For an LP-type problem (4) with pseudo-convex residuals, the maximum size of a basis, which we call *combinatorial dimension*, is $d + 1$.

Definition 2 (Violation set, level and coverage). *The violation set of a basis $\mathcal{B}$ is defined as $\mathcal{V}(\mathcal{B}) = \{\mathbf{s}_i \in \mathcal{S} | \ r(\boldsymbol{\theta}(\mathcal{B})|\mathbf{s}_i) > f(\mathcal{B})\}$. We call $l(\mathcal{B}) = |\mathcal{V}(\mathcal{B})|$ the level of $\mathcal{B}$ and $\mathcal{C}(\mathcal{B}) = \mathcal{S}\backslash\mathcal{V}(\mathcal{B})$ the coverage of $\mathcal{B}$.*

By the above definition,

$$c(\boldsymbol{\theta}(\mathcal{B})|\mathcal{S}) = |\mathcal{S}| - l(\mathcal{B}). \tag{5}$$

An important property of LP-type problems is that solving (4) on $\mathcal{C}(\mathcal{B})$ and $\mathcal{B}$ return the same solution.

Definition 3 (Support set). *The level-0 basis for $\mathcal{S}$ is called the support set of $\mathcal{S}$, which we represent as $\tau(\mathcal{S})$.*

Assume we know the maximal inlier set $\mathcal{I}$ for (1), where $|\mathcal{I}| = c(\boldsymbol{\theta}^*|\mathcal{S})$. Define $\mathcal{B}^* = \tau(\mathcal{I})$ as the support set of $\mathcal{I}$; $\mathcal{B}^*$ can be obtained by solving (4) on $\mathcal{I}$. Then, $l(\mathcal{B}^*)$ is the size of the minimal outlier set. Our target problem (1) can then be recast as finding the optimal basis

$$\mathcal{B}^* = \underset{\mathcal{B}\subseteq\mathcal{S}}{\text{argmin}} \ l(\mathcal{B}), \ \text{s.t.} \ f(\mathcal{B}) \leq \epsilon, \tag{6}$$

and $\boldsymbol{\theta}(\mathcal{B}^*)$ is the maximizer of (1). Intuitively, $\mathcal{B}^*$ is the lowest level basis that is feasible, where a basis $\mathcal{B}$ is called feasible if $f(\mathcal{B}) \leq \epsilon$.

2.3. A* tree search algorithm

Matoušek [18] showed that the set of bases for an LP-type problem can be arranged in a tree, where the root node is $\tau(\mathcal{S})$, and the level occupied by a node $\mathcal{B}$ on the tree is $l(\mathcal{B}) = |\mathcal{V}(\mathcal{B})|$. Another key insight is that there exists a path from $\tau(\mathcal{S})$ to any higher level basis, where a path is formed by a sequence of *adjacent bases*, defined as follows.

Algorithm 1 A* tree search of Chin et al. [6] for (6)

Require: $\mathcal{S} = \{\mathbf{s}_i\}_{i=1}^N$, threshold ϵ.
1: Insert $\mathcal{B} = \tau(\mathcal{S})$ with priority $e(\mathcal{B})$ into queue q.
2: Initialize hash table T to NULL.
3: **while** q is not empty **do**
4: Retrieve from q the $\mathcal{B}$ with the lowest $e(\mathcal{B})$.
5: **if** $f(\mathcal{B}) \leq \epsilon$ **then**
6: return $\mathcal{B}^* = \mathcal{B}$.
7: **end if**
8: $\mathcal{B}_r \leftarrow$ Attempt to reduce $\mathcal{B}$ by TOD method.
9: **for** each $\mathbf{s} \in \mathcal{B}_r$ **do**
10: **if** indices of $\mathcal{V}(\mathcal{B}) \cup \{\mathbf{s}\}$ do not exist in T **then**
11: Hash indices of $\mathcal{V}(\mathcal{B}) \cup \{\mathbf{s}\}$ into T.
12: $\mathcal{B}' \leftarrow \tau(\mathcal{C}(\mathcal{B})\backslash\{\mathbf{s}\})$.
13: Insert $\mathcal{B}'$ with priority $e(\mathcal{B}')$ into q.
14: **end if**
15: **end for**
16: **end while**
17: Return error (no inlier set of size greater than p).

Algorithm 2 Admissible heuristic h_{ins} for A* tree search

Require: $\mathcal{B}$
1: If $f(\mathcal{B}) \leq \epsilon$, return 0.
2: $\mathcal{O} \leftarrow \emptyset$.
3: **while** $f(\mathcal{B}) > \epsilon$ **do**
4: $\mathcal{O} \leftarrow \mathcal{O} \cup \mathcal{B}$, $\mathcal{B} \leftarrow \tau(\mathcal{C}(\mathcal{B})\backslash\mathcal{B})$.
5: **end while**
6: $h_{ins} \leftarrow 0$, $\mathcal{F} \leftarrow \mathcal{C}(\mathcal{B})$.
7: **for** each $\mathcal{B} \in \mathcal{O}$ **do**
8: **for** each $\mathbf{s} \in \mathcal{B}$ **do**
9: $\mathcal{B}' \leftarrow \tau(\mathcal{F} \cup \{\mathbf{s}\})$.
10: **if** $f(\mathcal{B}') \leq \epsilon$ **then**
11: $\mathcal{F} \leftarrow \mathcal{F} \cup \{\mathbf{s}\}$.
12: **else**
13: $h_{ins} \leftarrow h_{ins} + 1$, $\mathcal{F} \leftarrow \mathcal{F} \cup \{\mathbf{s}\}\backslash\mathcal{B}'$.
14: **end if**
15: **end for**
16: **end for**
17: **return** h_{ins}.

Definition 4 (Basis adjacency). *Two bases $\mathcal{B}'$ and $\mathcal{B}$ are adjacent if $\mathcal{V}(\mathcal{B}') = \mathcal{V}(\mathcal{B}) \cup \{\mathbf{s}_i\}$ for some $\mathbf{s}_i \in \mathcal{B}$.*

Intuitively, $\mathcal{B}'$ is a direct child of $\mathcal{B}$ in the tree. We say that we "follow the edge" from $\mathcal{B}$ to $\mathcal{B}'$ when we compute $\tau(\mathcal{C}(\mathcal{B})\backslash\{\mathbf{s}_i\})$. Chin et al. [6] solve (6) by searching the tree structure using the A* shortest path finding technique (Algorithm 1). Given input data $\mathcal{S}$, A* tree search starts from the root node $\tau(\mathcal{S})$ and iteratively expands the tree until $\mathcal{B}^*$ is found. The queue q stores all unexpanded tree nodes. And in each iteration, a basis $\mathcal{B}$ with the lowest *evaluation* value $e(\mathcal{B})$ is expanded. The expansion follows the basis adjacency, which computes $\tau(\mathcal{C}(\mathcal{B})\backslash\{\mathbf{s}\})$ for all $\mathbf{s} \in \mathcal{B}$ (Line 12 in Algorithm 1).

The evaluation value is defined as

$$e(\mathcal{B}) = l(\mathcal{B}) + h(\mathcal{B}), \tag{7}$$

where $h(\mathcal{B})$ is a *heuristic* which estimates the number of outliers in $\mathcal{C}(\mathcal{B})$. A* search uses only *admissible* heuristics.

Definition 5 (Admissibility). *A heuristic h is admissible if $h(\mathcal{B}) \geq 0$ and $h(\mathcal{B}) \leq h^*(\mathcal{B})$, where $h^*(\mathcal{B})$ is the minimum number of data that must removed from $\mathcal{C}(\mathcal{B})$ to make the remaining data feasible.*

Note that setting $e(\mathcal{B}) = l(\mathcal{B})$ (i.e., $h(\mathcal{B}) = 0$) for all $\mathcal{B}$ reduces A* search to breadth-first search (BFS). With an admissible heuristic, A* search is guaranteed to always find $\mathcal{B}^*$ before other sub-optimal feasible bases (see [6] for the proof). Algorithm 2 describes the heuristic h_{ins} used in [6].

Intuitively, the algorithm for h_{ins} removes a sequence of bases in the first round of iteration until a feasible subset $\mathcal{F} \subseteq \mathcal{C}(\mathcal{B})$ is found. After that, the algorithm iteratively inserts each removed basis point $\mathbf{s}$ back into $\mathcal{F}$. If the insertion of $\mathbf{s}$ makes $\mathcal{F}$ infeasible, $\tau(\mathcal{F} \cup \{\mathbf{s}\})$ is removed from the expanded $\mathcal{F}$ and the heuristic value h_{ins} is increased by 1.

The admissibility of h_{ins} is proved in [6, Theorem 4]. In brief, denote $\mathcal{F}^*$ as the largest feasible subset of $\mathcal{C}(\mathcal{B})$. If $\mathcal{F} \cup \{\mathbf{s}\}$ is infeasible, $\tau(\mathcal{F} \cup \{\mathbf{s}\})$ must contain at least one point in $\mathcal{F}^*$. Since we only add 1 to h_{ins} when this happens, then $h^*(\mathcal{B}) \geq h_{ins}(\mathcal{B})$.

2.4. Avoiding redundant node expansions

Algorithm 1 employs two strategies to avoid redundant node expansions. In Line 8, before expanding $\mathcal{B}$, a fast heuristic called True Outlier Detection (TOD) [6] is used to attempt to identify and remove true outliers from $\mathcal{B}$ (more details in Sec. 4), which has the potential to reduce the size of the branch starting from $\mathcal{B}$. In Line 10, a repeated basis check heuristic is performed to prevent bases that have been explored previously to be considered again (details in Sec. 3).

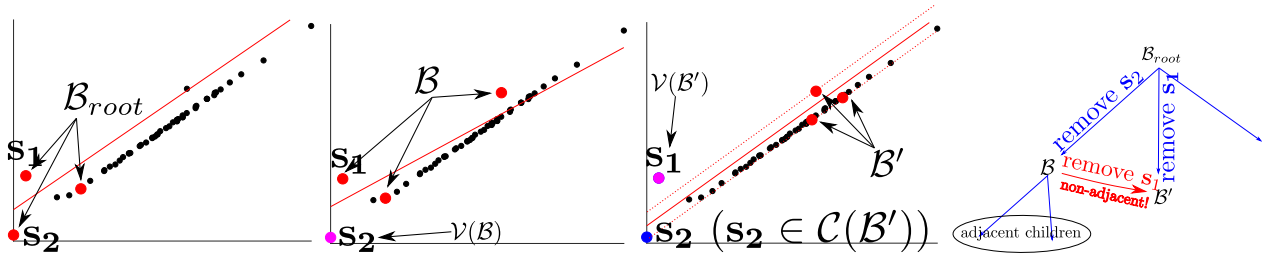
Our main contributions are two new strategies that improve upon the original methods above, as we will describe in Secs. 3 and 4. In each of the sections, we will first carefully analyze the weaknesses of the existing strategies. Sec. 5 will then put our new strategies in an overall algorithm. Sec. 6 presents the results.

3. Non-adjacent path avoidance

Recall Definition 4 on adjacency: for $\mathcal{B}$ and $\mathcal{B}'$ to be adjacent, their violation sets $\mathcal{V}(\mathcal{B})$ and $\mathcal{V}(\mathcal{B}')$ must differ by one point; in other words, it must hold that

$$|l(\mathcal{B}') - l(\mathcal{B})| = 1. \tag{8}$$

Given a $\mathcal{B}$, Line 12 in Algorithm 1 generates an adjacent "child" basis of $\mathcal{B}$ by removing a point $\mathbf{s}$ from $\mathcal{B}$ and solving



(a) Root node $\mathcal{B}_{root}$. (b) Level-1 node $\mathcal{B}$. (c) Level-1 node $\mathcal{B}$. $\mathbf{s}_2 \in \mathcal{C}(\mathcal{B})$. (d) Tree structure

Figure 1. (a–c) Path between non-adjacent bases ($\mathcal{B} \to \mathcal{B}'$). $\mathcal{B}'$ can be generated from both $\mathcal{B}$ and $\mathcal{B}_{root}$, but it is *not* adjacent to $\mathcal{B}$ since $l(\mathcal{B}') = l(\mathcal{B})$. Note that Line 10 in Algorithm 1 cannot avoid this non-adjacent path since $\mathcal{V}(\mathcal{B}') \cup \{\mathbf{s}_2\} = \{\mathbf{s}_1, \mathbf{s}_2\} \neq \mathcal{V}(\mathcal{B}') = \{\mathbf{s}_1\}$. Panel (d) shows the relationship between the three bases during tree search. In the proposed Non-Adjacent Path Avoidance (NAPA) strategy, the path drawn in red is not followed. As we will show in Sec. 6, this simple idea provides a massive reduction in runtime of A* tree search.

the minimax problem (4) on $\mathcal{C}(\mathcal{B})\backslash\{\mathbf{s}\}$. In this way,

$$l(\mathcal{B}') = l(\mathcal{B}) + 1. \tag{9}$$

Iterating the $\mathbf{s}$ to be removed thus generates all the adjacent child bases of $\mathcal{B}$, which allows the tree to be explored.

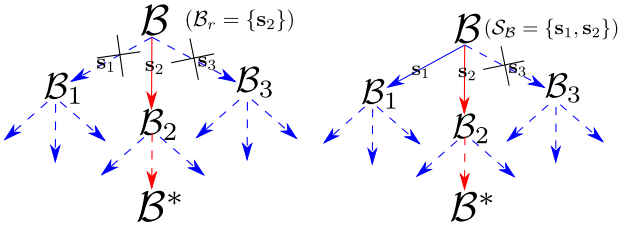
However, an important phenomenon that is ignored in Algorithm 1 is, while the above process generates all the adjacent child bases of $\mathcal{B}$, *not all $\mathcal{B}'$ generated in the process are adjacent child bases*. Figure 1 shows a concrete example from line fitting (2): from a root node $\mathcal{B}_{root}$, two child bases $\mathcal{B}$ and $\mathcal{B}'$ are generated by respectively removing points $\mathbf{s}_2$ and $\mathbf{s}_1$. However, by further removing $\mathbf{s}_1$ from $\mathcal{B}$ and solving (4) on $\mathcal{C}(\mathcal{B} \setminus \{\mathbf{s}_1\})$, we obtain $\mathcal{B}'$ again! Since $l(\mathcal{B}') = l(\mathcal{B})$, these two bases are not adjacent.

In general, non-adjacent paths occur in Algorithm 1 when some elements of $\mathcal{V}(\mathcal{B})$ are in $\mathcal{C}(\mathcal{B}')$ after solving the minimax problem on $\mathcal{C}(\mathcal{B} \setminus \{\mathbf{s}\})$. While inserting a non-adjacent $\mathcal{B}'$ into the queue does not affect global optimality, it does reduce efficiency. This is because the repeated basis check heuristic in Algorithm 1 assumes that the level of the child node $\mathcal{B}'$ is always lower than the parent $\mathcal{B}$ by 1; this assumption does not hold if the generated basis $\mathcal{B}'$ is not adjacent. More formally, if $\mathcal{B}'$ is not adjacent to $\mathcal{B}$, then

$$\mathcal{V}(\mathcal{B}) \cup \{\mathbf{s}\} \neq \mathcal{V}(\mathcal{B}') \tag{10}$$

and the repeated basis check in Line 8 in Algorithm 1 fails. Since the same $\mathcal{B}'$ could be generated from its "real" parent (e.g., in Figure 1, $\mathcal{B}'$ was also generated by $\mathcal{B}_{root}$), the same basis can be inserted into the queue more than once.

Since tree search only needs adjacent paths, we can safely skip traversing any non-adjacent path without affecting the final solution. To do this, we propose a Non-Adjacent Path Avoidance (NAPA) strategy for A* tree search; see Fig. 1(d). Given a basis $\mathcal{B}$, any non-adjacent basis generated from it cannot have a level that is higher than $l(\mathcal{B})$. Therefore, we can simply discard any newly generated basis $\mathcal{B}'$ (Line 12) if $l(\mathcal{B}') \leq l(\mathcal{B})$. Though one redundant minimax problem (4) still needs to be solved when finding $\mathcal{B}'$, a much larger cost for computing $e(\mathcal{B}')$ (which requires to solve multiple problems (4)) is saved along with all the computation required for traversing the children of $\mathcal{B}'$. The effectiveness of this strategy will be demonstrated later in Sec. 6.



(a) TOD (b) DIBP

Figure 2. (a) In TOD, on current node $\mathcal{B}$, if $\mathbf{s}_2$ is identified as the true outlier, then the shortest path towards a feasible basis $\mathcal{B}^*$ must pass through $\mathbf{s}_2$ (path rendered in red). All the the other $|\mathcal{B}| - 1$ branches (leading from $\mathbf{s}_1$ and $\mathbf{s}_3$ in this example) can be skipped. (b) In DIBP, instead of attempting to identify a single true outlier, a group $\mathcal{S}_\mathcal{B}$ that contains at least one true outlier ($\mathcal{S}_\mathcal{B} = \{\mathbf{s}_1, \mathbf{s}_2\}$ in this example) is identified; if this is successful, the other $|\mathcal{B}| - |\mathcal{S}_\mathcal{B}|$ paths (corresponding to $\mathbf{s}_3$ in this example) can be skipped. DIBP is more effective than TOD because it is easier to reject a subset than a single point as outlier; see Sec. 4.2 for details.

4. Dimension-insensitive branch pruning

Our second improvement to A* tree search lies in a new branch pruning technique. We first review the original method (TOD) and then describe our new technique.

4.1. Review of true outlier detection (TOD)

Referring to Line 8 in Algorithm 1 [6], let $\mathcal{F}^*$ be the largest feasible subset of $\mathcal{C}(\mathcal{B})$. A point $\mathbf{s} \in \mathcal{B}$ is said to be a true outlier if $\mathbf{s} \notin \mathcal{F}^*$, otherwise we call it a true inlier. Given an infeasible node $\mathcal{B}$, one of the elements in $\mathcal{B}$ must be a true outlier. The goal of TOD is to identify one such true outlier in $\mathcal{B}$. If $\mathbf{s} \in \mathcal{B}$ is successfully identified as a

true outlier, we can skip the child generation for all the other points in $\mathcal{B}$ without hurting optimality, since $\mathbf{s}$ must be on the shortest path to feasibility via $\mathcal{B}$; see Fig. 2(a). If such an $\mathbf{s}$ can be identified, the reduced subset $\mathcal{B}_r$ is simply $\{\mathbf{s}\}$.

The principle of TOD is as follows: define $h^*(\mathcal{B}|\mathbf{s})$ as the *minimum* number of data points that must be removed from $\mathcal{C}(\mathcal{B})$ to achieve feasibility, with $\mathbf{s}$ forced to be feasible. We can conclude that $\mathbf{s} \in \mathcal{B}$ is a true outlier if and only if

$$h^*(\mathcal{B}|\mathbf{s}) > h^*(\mathcal{B}); \tag{11}$$

see [6] for the formal proof. Intuitively, if $\mathbf{s}$ is a true inlier, forcing its feasibility will not change the value of h^*. On the other hand, if forcing $\mathbf{s}$ to be feasible leads to the above condition, $\mathbf{s}$ cannot be a true inlier.

Bound computation for TOD. Unsurprisingly $h^*(\mathcal{B}|\mathbf{s})$ is as difficult to compute as $h^*(\mathcal{B})$. To avoid directly computing $h^*(\mathcal{B}|\mathbf{s})$, TOD computes an admissible heuristic $h(\mathcal{B}|\mathbf{s})$ of $h^*(\mathcal{B}|\mathbf{s})$ and an upper bound $g(\mathcal{B})$ of $h^*(\mathcal{B})$. Given $\mathbf{s} \in \mathcal{B}$, $h(\mathcal{B}|\mathbf{s})$ and $g(\mathcal{B})$, if

$$h(\mathcal{B}|\mathbf{s}) > g(\mathcal{B}), \tag{12}$$

then it must hold that

$$h^*(\mathcal{B}|\mathbf{s}) \geq h(\mathcal{B}|\mathbf{s}) > g(\mathcal{B}) \geq h^*(\mathcal{B}), \tag{13}$$

which implies that $\mathbf{s}$ is a true outlier.

As shown in [6], $g(\mathcal{B})$ can be computed as a by-product of computing $h_{ins}(\mathcal{B})$, and $h(\mathcal{B}|\mathbf{s})$ is computed by a constrained version of h_{ins}, which we denote as $h_{ins}(\mathcal{B}|\mathbf{s})$. Computing $h_{ins}(\mathcal{B}|\mathbf{s})$ is done by the constrained version of Algorithm 2, where all minimax problems (4) required to solve are replaced by their constrained versions, which are in the following form:

$$\underset{\boldsymbol{\theta}}{\text{minimize}} \quad \max_{\mathbf{s}_i \in \mathcal{S}^1} r(\boldsymbol{\theta}|\mathbf{s}_i), \tag{14a}$$

$$\text{s.t.} \quad r(\boldsymbol{\theta}|\mathbf{s}'_j) \leq \epsilon, \ \forall \mathbf{s}'_j \in \mathcal{S}'. \tag{14b}$$

The only difference between (14) and (4) is the constraint that all data in $\mathcal{S}'$ must be feasible. And similar to (4), (14) is also an LP-type problem which can be solved by standard solvers [10]. Similar as in (4) we also define $f(\mathcal{S}^1|\mathcal{S}')$ as the minimum objective value of (14) and $\boldsymbol{\theta}(\mathcal{S}^1|\mathcal{S}')$ as the corresponding optimal solution.

With the above definition, changing Algorithm 2 to its constrained version can be simply done by replacing $f(\mathcal{B})$ (Line 3) and $f(\mathcal{B}')$ (Line 10) by $f(\mathcal{B}|\{\mathbf{s}\})$ and $f(\mathcal{B}'|\{\mathbf{s}\})$.

Why is TOD ineffective? The effectiveness of TOD in accelerating Algorithm 1 depends on how frequent TOD can detect a true outlier. When a true outlier for $\mathcal{B}$ is detected, TOD prunes $|\mathcal{B}|-1$ branches; on the flipside, if TOD cannot identify an $\mathbf{s} \in \mathcal{B}$ as the true outlier, the runtime to compute $h_{ins}(\mathcal{B}|\mathbf{s})$ will be wasted. In the worst case where no true outlier is identified for $\mathcal{B}$, Algorithm 2 has to be executed redundantly for $|\mathcal{B}|$ times. Whether TOD can find the true outlier is largely decided by how well $h_{ins}(\mathcal{B}|\mathbf{s})$ approximates $h^*(\mathcal{B}|\mathbf{s})$.

We now show that $h_{ins}(\mathcal{B}|\mathbf{s})$ is usually a poor estimator of $h^*(\mathcal{B}|\mathbf{s})$. Define $\mathcal{O}^*(\mathcal{B}|\mathbf{s})$ as the smallest subset that must be removed from $\mathcal{C}(\mathcal{B})\backslash\mathbf{s}$ to achieve feasibility, with $\mathbf{s}$ forced to be feasible, i.e., $|\mathcal{O}^*(\mathcal{B}|\mathbf{s})| = h^*(\mathcal{B}|\mathbf{s})$. Then, $h_{ins}(\mathcal{B}|\mathbf{s})$ and $h^*(\mathcal{B}|\mathbf{s})$ will be different if a basis $\mathcal{B}_{rem}$ removed during Algorithm 2 contains multiple elements in $\mathcal{O}^*(\mathcal{B}|\mathbf{s})$, since we only add 1 to h_{ins} when actually more than 1 points in $\mathcal{B}_{rem}$ should be removed. And the following lemma shows that the difference between $h_{ins}(\mathcal{B}|\mathbf{s})$ and $h^*(\mathcal{B}|\mathbf{s})$ will be too large for TOD to be effective if the rate of true outliers in $\mathcal{C}(\mathcal{B})$, i.e., $\frac{h^*(\mathcal{B})}{\mathcal{C}(\mathcal{B})}$, is too large.

Lemma 1. *Condition* (12) *is always false when*

$$\frac{h^*(\mathcal{B})}{\mathcal{C}(\mathcal{B})} \geq \frac{1}{\phi} \cdot \frac{|\mathcal{C}(\mathcal{B})|-1}{|\mathcal{C}(\mathcal{B})|}, \tag{15}$$

where ϕ is the average *size of all $\mathcal{B}_{rem}$ during Algorithm 2.*

Proof. Since $h_{ins}(\mathcal{B}|\mathbf{s})$ is the number of $\mathcal{B}_{rem}$ during Algorithm 2, $h_{ins}(\mathcal{B}|\mathbf{s}) \cdot \phi \leq |\mathcal{C}(\mathcal{B})\backslash\{\mathbf{s}\}| = |\mathcal{C}(\mathcal{B})| - 1$. Hence,

$$h_{ins}(\mathcal{B}|\mathbf{s}) \leq \frac{|\mathcal{C}(\mathcal{B})|-1}{\phi}, \tag{16}$$

Therefore, condition (12) can never be true if

$$h^*(\mathcal{B}) \geq \frac{|\mathcal{C}(\mathcal{B})|-1}{\phi}. \tag{17}$$

Dividing both sides of (17) by $\mathcal{C}(\mathcal{B})$ leads to (15). □

Intuitively, when (15) happens, there are too many outliers in $\mathcal{C}(\mathcal{B})$ hence too many $\mathcal{B}_{rem}$ that include multiple elements in $\mathcal{O}^*(\mathcal{B}|\mathbf{s})$, making $h_{ins}(\mathcal{B}|\mathbf{s})$ too far from $h^*(\mathcal{B}|\mathbf{s})$.

In addition, ϕ is positively correlated with d, and in the worst case can be $d+1$, which makes TOD sensitive to d. Figure 3 shows the effectiveness of TOD as a function of d, for problems with linear residual (2). As can be seen, the outlier rate where TOD can be effective reduces quickly with d ($<15\%$ when $d \geq 7$). Note that since $g(\mathcal{B})$ is only an estimation of $h^*(\mathcal{B})$, the actual range where TOD is effective can be smaller than the region above the dashed line.

4.2. New pruning technique: DIBP

Due to the above limitation, TOD is often not effective in pruning; the cost to carry out Line 8 in Algorithm 1 is thus usually wasted. To address this issue, we propose a more effective branch pruning technique called DIBP (dimension-insensitive branch pruning).

DIBP extends the idea of TOD, where instead of searching for one true outlier, we search for a *subset* $\mathcal{S}_\mathcal{B}$ of $\mathcal{B}$ that

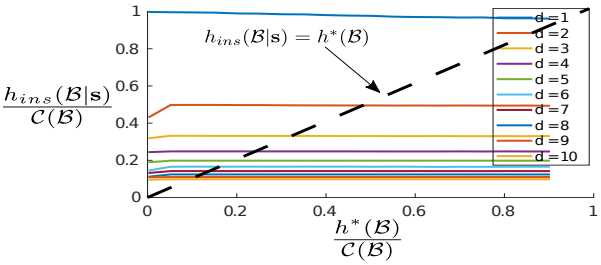


Figure 3. Effectiveness of TOD as a function of d. All problem instances are generated randomly and each solid curve contains data with true outlier rates $\frac{h^*(\mathcal{B})}{\mathcal{C}(\mathcal{B})}$ from 0 to 90%. Note that (15) is true for a d when the solid curve for the d is below the dashed line.

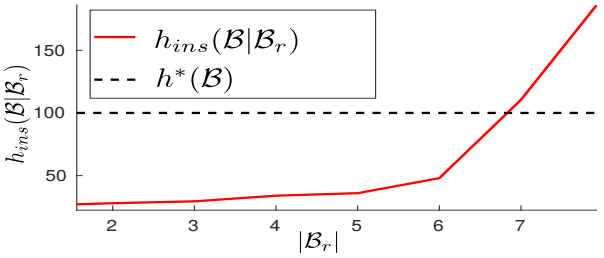


Figure 4. Effectiveness of DIBP when $d = 8$. $|\mathcal{C}(\mathcal{B})| = 200$. $h_{ins}(\mathcal{B}|\mathcal{S}_\mathcal{B})$ increases stably along with $|\mathcal{S}_\mathcal{B}|$ and is effective even when the true outlier rate is 90%. Though only the 50% case is shown, changing the outlier rate in practice merely affects the values of $h_{ins}(\mathcal{B}|\mathcal{S}_\mathcal{B})$ as long as the data distribution is similar.

must contain at least one true outlier. If such a subset can be identified, the children of $\mathcal{B}$ corresponding to removing points not in $\mathcal{S}_\mathcal{B}$ can be ignored during node expansion—again, this is because the shortest path to feasibility via $\mathcal{B}$ must go via $\mathcal{S}_\mathcal{B}$; Fig. 2(b) illustrates this idea.

To find such an $\mathcal{S}_\mathcal{B}$, we greedily add points from $\mathcal{B}$ into $\mathcal{S}_\mathcal{B}$ to see whether enforcing the feasibility of $\mathcal{S}_\mathcal{B}$ contradicts the following inequality

$$h_{ins}(\mathcal{B}|\mathcal{S}_\mathcal{B}) > g(\mathcal{B}), \tag{18}$$

which is the extension of (12), with $h = h_{ins}$. Similar to $h_{ins}(\mathcal{B}|\mathbf{s})$, $h_{ins}(\mathcal{B}|\mathcal{S}_\mathcal{B})$ is computed by the constrained version of Algorithm 2 with $\mathcal{S}' = \mathcal{S}_\mathcal{B}$ in problem (14).

The insight is that by adding more and more constraints into problem (14), the average basis size ϕ will gradually reduce, making the right hand side of (15) increase until it exceeds the left hand side, so that even with large d, branch pruning will be effective with high true outlier rate. Figure 4 shows the effectiveness of DIBP for an 8-dimensional problem with linear residuals. Observe that $h_{ins}(\mathcal{B}|\mathcal{S}_\mathcal{B})$ increases steadily along with $|\mathcal{S}_\mathcal{B}|$ and can tolerate more than 90% of true outliers when $|\mathcal{S}_\mathcal{B}| = |\mathcal{B}| - 1 = 8$.

During DIBP, we want to add true outliers into $\mathcal{S}_\mathcal{B}$ as soon as possible, since (18) can never be true if $\mathcal{S}_\mathcal{B}$ contains no true outliers. To do so, we utilize the corresponding solution $\boldsymbol{\theta}_{g(\mathcal{B})}$ that leads to $g(\mathcal{B})$. During DIBP, the $\mathbf{s} \in \mathcal{B}$ with the largest residual $r(\boldsymbol{\theta}_{g(\mathcal{B})}|\mathbf{s})$ will be added into $\mathcal{S}_\mathcal{B}$ first, since a larger residual means a higher chance that $\mathbf{s}$ is a true outlier. In practice, this strategy often enables DIBP to find close to minimal-size $\mathcal{S}_\mathcal{B}$.

For problems with linear residuals, we can further compute an adaptive starting value $z(\mathcal{B})$ of $|\mathcal{S}_\mathcal{B}|$, where DIBP can safely skip the first $z(\mathcal{B})-1$ computations of $h_{ins}(\mathcal{B}|\mathcal{S}_\mathcal{B})$ without affecting the branch pruning result. The value of $z(\mathcal{B})$ should be $\max\{1, d+2-\frac{|\mathcal{C}(\mathcal{B})|-1}{g(\mathcal{B})}\}$. The reason is demonstrated in the following lemma:

Lemma 2. *For problems with linear residuals,* (18) *cannot be true unless*

$$|\mathcal{S}_\mathcal{B}| > d + 1 - \frac{|\mathcal{C}(\mathcal{B})| - 1}{g(\mathcal{B})}. \tag{19}$$

Proof. As in (16), we have $h_{ins}(\mathcal{B}|\mathcal{S}_\mathcal{B}) < \frac{|\mathcal{C}(\mathcal{B})|-1}{\phi}$. To ensure that (18) can be true, we must have $g(\mathcal{B}) < \frac{|\mathcal{C}(\mathcal{B})|-1}{\phi}$, which we rewrite as

$$\phi < \frac{|\mathcal{C}(\mathcal{B})| - 1}{g(\mathcal{B})}. \tag{20}$$

And for problems with linear residuals, (14) with $\mathcal{S}' = \mathcal{S}_\mathcal{B}$ is a linear program, whose optimal solution resides at a vertex of the feasible polytope [19, Chapter 13]. This means that for problem (14), the basis size plus the number of active constraints at the optimal solution must be $d + 1$. And since each absolute-valued constraint in (14b) can at most contribute one active linear constraint, the maximum number of active constraints is $|\mathcal{S}_\mathcal{B}|$. Thus during the computation of $h_{ins}(\mathcal{B}|\mathcal{S}_\mathcal{B})$, the average basis size $\phi \geq d + 1 - |\mathcal{S}_\mathcal{B}|$. Substituting this inequality into (20) results in (19). □

5. Main algorithm

Algorithm 3 summarizes the A* tree search algorithm with our new acceleration techniques. A reordering is done so that cheaper acceleration techniques are executed first. Specifically, given the current basis $\mathcal{B}$, we iterate through each element $\mathbf{s} \in \mathcal{B}$ and check first whether it leads to a repeated adjacent node and skip $\mathbf{s}$ if yes (Line 8). Otherwise, we check whether the node $\mathcal{B}'$ generated by $\mathbf{s}$ is non-adjacent to $\mathcal{B}$ and discard $\mathcal{B}'$ if yes (Line 11). If not, we insert $\mathcal{B}'$ into the queue since it cannot be pruned by other techniques. After that, we perform DIBP (Line 14) and skip the other elements in $\mathcal{B}$ if condition (18) is satisfied. Note that we can still add $\mathbf{s}$ into $\mathcal{S}_\mathcal{B}$ even though it leads to repeated bases. This strategy makes DIBP much more effective in practice.

6. Experiments

To demonstrate the effectiveness of our new techniques, we compared the following A* tree search variants:

Algorithm 3 A* tree search with NAPA and DIBP

Require: $\mathcal{S} = \{\mathbf{s}_i\}_{i=1}^{N}$, threshold ϵ.
Insert $\mathcal{B} = \tau(\mathcal{S})$ with priority $e(\mathcal{B})$ into queue q.
Initialize hash table T to NULL.
while q is not empty **do**
  Retrieve from q the $\mathcal{B}$ with the lowest $e(\mathcal{B})$.
  If $f(\mathcal{B}) \leq \epsilon$ **then** return $\mathcal{B}^* = \mathcal{B}$.
  $\mathcal{S}_\mathcal{B} \leftarrow \emptyset$; Sort $\mathcal{B}$ descendingly based on $r(\boldsymbol{\theta}_{g(\mathcal{B})}|\mathbf{s})$.
  for each $\mathbf{s} \in \mathcal{B}$ **do**
    if indices of $\mathcal{V}(\mathcal{B}) \cup \{\mathbf{s}\}$ do not exist in T **then**
      Hash indices of $\mathcal{V}(\mathcal{B}) \cup \{\mathbf{s}\}$ into T.
      $\mathcal{B}' \leftarrow \tau(\mathcal{C}(\mathcal{B}) \backslash \{\mathbf{s}\})$.
      if $l(\mathcal{B}') > l(\mathcal{B})$ **then**.
        $\mathcal{S}_\mathcal{B} \leftarrow \mathcal{S}_\mathcal{B} \cup \{\mathbf{s}\}$.
        Insert $\mathcal{B}'$ with priority $e(\mathcal{B}')$ into q.
        If $|\mathcal{S}_\mathcal{B}| = |\mathcal{B}| \vee$ (18) is true **then** break.
      end if
    else
      $\mathcal{S}_\mathcal{B} \leftarrow \mathcal{S}_\mathcal{B} \cup \{\mathbf{s}\}$.
    end if
  end for
end while
Return error (no inlier set of size greater than p).

- Original A* tree search (A*) [5].
- A* with TOD for branch pruning (A*-TOD) [6].
- A* with non-adjacenct path avoidance (A*-NAPA).
- A*-NAPA with TOD branch pruning (A*-NAPA-TOD).
- A*-NAPA with DIBP branch pruning (A*-NAPA-DIBP).

All variants were implemented in MATLAB 2018b, based on the original code of A*. For problems with linear residuals, we use the self-implemented vertex-to-vertex algorithm [3] to solve the minimax problems (4) and (14). And in the non-linear case, these two problems were solved by the matlab function `fminimax`. All experiments were executed on a laptop with Intel Core 2.60GHz i7 CPU, 16GB RAM and Ubuntu 14.04 OS.

6.1. Controlled experiment on synthetic data

To analyze the effect of o and N to different methods, we conducted a controlled experiment on the 8-dimensional robust linear regression problem with different N and o. The residual of linear regression is in the form of (2). To generate data $\mathcal{S} = \{\mathbf{a}_i, b_i\}_{i=1}^{N}$, a random model $\boldsymbol{\theta} \in \mathbb{R}^d$ was first generated and N data points that perfectly fit the model were randomly sampled. Then, we randomly picked $N - o$ points as inliers and assigned to the b_i of these points noise uniformly distributed between $[-0.1, 0.1]$. Then we assigned to the other o points noise uniformly distributed from $[-5, -0.1) \cup (0.1, 5]$ to create a controlled number of outliers. The inlier threshold ϵ was set to 0.1.

To verify the superior efficiency of tree search compared to other types of globally optimal methods, we also tested the Mixed Integer Programming-based BnB algorithm (MIP) [28] in this experiment. The state-of-the-art Gurobi solver was used as the optimizer for MIP. MIP was parallelized by Gurobi using 8 threads, while all tree search methods were executed sequentially.

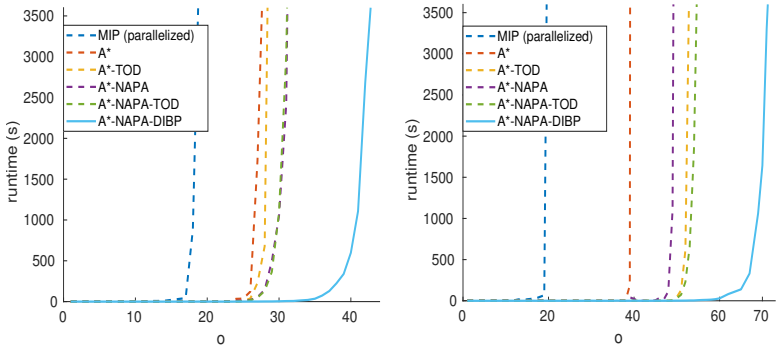
As shown in Figure 5, all A* tree search variants are much faster than MIP, even though MIP was significantly accelerated by parallel computing. Both NAPA and DIBP brought considerable acceleration to A* tree search, which can be verified by the gaps between the variants with and without these techniques. Note that when $N = 200$, A*-NAPA had similar performance with and without TOD, while DIBP provided stable and significant acceleration for all data.

Interestingly, having a larger N made A* tree search efficient for a much larger o. This can be explained by condition (15). With the same o, a larger N meant a lower true outlier rate, which made (15) less likely.

6.2. Linearized fundamental matrix estimation

Experiments were also conducted on real data. We executed all tree seach variants for linearized fundamental matrix estimation [6], which used the algebaric error [13, Sec.11.3] as the residual and ignored the non-convex rank-2 constraints. 5 image pairs (the first 5 crossroads) were selected from the sequence `00` of the KITTI Odometry dataset [12]. For each image pair, the input was a set of SIFT [17] feature matches generated using VLFeat [27]. The inlier threshold ϵ was set to 0.03 for all image pairs.

The result is shown in Table 1. We also showed the number of unique nodes (NUN) generated and the number of branch pruning steps (NOBP) executed before the termination of each algorithm. A*-NAPA-DIBP found the optimal solution in less than 10s for all data, while A* and A*-TOD often failed to finish in 2 hours. A*-NAPA-DIBP was faster by more than 500 times on all data compared to the fastest method among A* and A*-TOD. For the effectiveness of each technique, applying NAPA to A* often resulted in more than 10x acceleration. And applying DIBP further sped up A*-NAPA by more than 1000x on challenging data (e.g. `Frame-198-201`). This significant acceleration is because many elements in $\mathcal{S}_\mathcal{B}$ were the ones that led to redundant nodes, which made most non-redundant paths effectively pruned. TOD was much less effective than DIBP and introduced extra runtime to A*-NAPA on `Frame-104-108` and `Frame-198-201`. We also attached o_{LRS}, the *estimated* number of outliers returned from LO-RANSAC [8], which is an effective RANSAC variant. None of the LO-RANSAC results were optimal. A visualization of the tree search result is shown in Figure 6.



(a) $N = 200$ (b) $N = 400$

Figure 5. Runtime vs o for robust linear regression on synthetic data. $d = 8$.

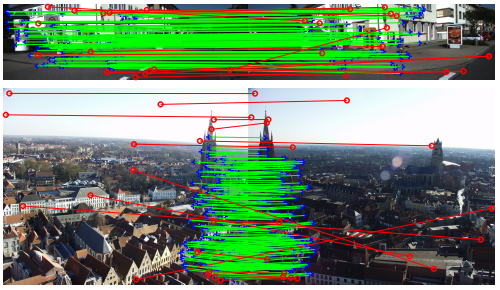


Figure 6. (Top) Fundamental matrix estimation result of A*-NAPA-DIBP on `Frame-738-742`. (Bottom) Homography estimation result of A*-NAPA-DIBP on data `BruggeTower`. The inliers (in green) in the top figure were down-sampled to 100 for clarity.

data $d = 8$	`Frame-104-108` $o = 13$ ($o_{LRS} = 23$); $N = 302$		`Frame-198-201` $o = 13$ ($o_{LRS} = 19$); $N = 309$		`Frame-417-420` $o = 19$ ($o_{LRS} = 23$); $N = 385$		`Frame-579-582` $o = 22$ ($o_{LRS} = 25$); $N = 545$		`Frame-738-742` $o = 14$ ($o_{LRS} = 32$); $N = 476$	
	NUN/NOBP	runtime (s)	NUN/NOBP	runtime (s)	NUN/NOBP	runtime (s)	NUN/NOBP	runtime (s)	NUN/NOBP	runtime (s)
A*	163232/0	> 6400	169369/0	> 6400	144560/0	> 6400	136627/0	> 6400	160756/0	> 6400
A*-TOD	134589/119871	> 6400	129680/126911	> 6400	80719/92627	3712.99	55764/58314	2709.21	49586/50118	1729.34
A*-NAPA	35359/0	561.81	23775/0	351.07	175806/0	5993.68	147200/0	> 6400	29574/0	471.15
A*-NAPA-TOD	33165/22275	770.08	19308/13459	451.39	15310/10946	429.06	15792/12073	576.82	14496/10752	373.36
A*-NAPA-DIBP	205/311	**7.63**	105/160	**3.88**	172/216	**6.85**	60/84	**3.49**	52/77	**2.00**
A*-NAPA-DIBP vs previous best method	best previous method A*/A*-TOD	faster by **> 839x**	best previous method A*/A*-TOD	faster by **> 1648x**	best previous method A*-TOD	faster by **541x**	best previous method A*-TOD	faster by **775x**	best previous method A*-TOD	faster by **864x**

Table 1. Linearized fundamental matrix estimation result. The names of the data are the image indices in the sequence. o_{LRS} is the *estimated* outlier number returned by LO-RANSAC. NUN: number of unique nodes generated. NOBP: number of branch pruning steps executed. The last row shows how much faster A*-NAPA-DIBP was, compared to the fastest previously proposed variants (A* and A*-TOD).

data $d = 8$	`Adam` $o = 38$ ($o_{LRS} = 40$); $N = 282$		`City` $o = 19$ ($o_{LRS} = 22$); $N = 87$		`Boston` $o = 43$ ($o_{LRS} = 44$); $N = 678$		`Brussels` $o = 9$ ($o_{LRS} = 25$); $N = 231$		`BruggeTower` $o = 17$ ($o_{LRS} = 26$); $N = 208$	
	NUN/NOBP	runtime (s)	NUN/NOBP	runtime (s)	NUN/NOBP	runtime (s)	NUN/NOBP	runtime (s)	NUN/NOBP	runtime (s)
A*	224/0	538.91	7072/0	> 6400	406/0	2455.03	397/0	437.25	5003/0	> 6400
A*-TOD	38/37	**156.98**	462/514	910.51	7/6	**74.63**	359/281	499.77	333/260	298.39
A*-NAPA	168/0	404.77	6481/0	> 6400	234/0	1284.14	264/0	268.85	3731/0	4740.68
A*-NAPA-TOD	38/37	**156.98**	286/241	485.36	7/6	**74.63**	249/191	297.91	201/151	161.95
A*-NAPA-DIBP	38/37	**156.98**	34/40	**64.44**	7/6	**74.63**	30/42	**50.13**	40/48	**68.20**
A*-NAPA-DIBP vs previous best method	best previous method A*-TOD	faster by same runtime	best previous method A*-TOD	faster by **13.1x**	best previous method A*-TOD	faster by same runtime	best previous method A*	faster by **7.7x**	best previous method A*-TOD	faster by **3.4x**

Table 2. Homography estimation result. o_{LRS} is the *estimated* outlier number returned by LO-RANSAC. NUN: number of unique nodes generated. NOBP: number of branch pruning steps executed. The last row shows how much faster A*-NAPA-DIBP was, compared to the fastest previously proposed variants (A* and A*-TOD).

6.3. Homography estimation (non-linear)

To test all methods on non-linear problems, another experiment for homography estimation [13] was done on "homogr" dataset[1]. As before, we picked 5 image pairs, computed the SIFT matches and used them as the input data. The transfer error in one image [13] was used as the residual, which was in the form of (3). ϵ was set to 4 pixels.

Table 2 shows the result of all methods. Compared to the linear case, solving non-linear minimax problems (4) and (14) was much more time-consuming (can be 100x slower with `fminimax`). Thus with similar NUN and NOBP, the runtime was much larger. However, the value of ϕ in the non-linear case was usually also much smaller, which made the heuristic h_{ins} and in turn all branch pruning techniques much more effective than in the linear case. And for easy data such as `Boston` and `Adam`, performing either TOD or DIBP was enough to achieve the highest speed. Nonetheless, DIBP was still much more effective than TOD on other data. And DIBP never slowed down the A* tree search as TOD sometimes did (e.g., in `Brussels`). A*-NAPA-DIBP remained fastest on all image pairs. An example of the visual result is provided in Figure 6.

[1] `http://cmp.felk.cvut.cz/data/geometry2view/index.xhtml`

7. Conclusion

We presented two new acceleration techniques for consensus maximization tree search. The first avoids redundant non-adjacent paths that exist in the consensus maximization tree structure. The second makes branch pruning much less sensitive to the problem dimension, and therefore much more reliable. The significant acceleration brought by the two techniques contributes a solid step towards practical and globally optimal consensus maximization.

Acknowledgements. We thank Dr. Nan Li for his valuable suggestions.

References

[1] Nina Amenta, Marshall Bern, and David Eppstein. Optimal point placement for mesh smoothing. *Journal of Algorithms*, 30(2):302–322, 1999.

[2] Zhipeng Cai, Tat-Jun Chin, Huu Le, and David Suter. Deterministic consensus maximization with biconvex programming. In *European Conference on Computer Vision (ECCV)*, 2018.

[3] E. W. Cheney. *Introduction to Approximation Theory*. McGraw-Hill, 1966.

[4] Tat-Jun Chin, Zhipeng Cai, and Frank Neumann. Robust fitting in computer vision: Easy or hard? In *European Conference on Computer Vision (ECCV)*, 2018.

[5] Tat-Jun Chin, Pulak Purkait, Anders Eriksson, and David Suter. Efficient globally optimal consensus maximisation with tree search. In *Computer Vision and Pattern Recognition (CVPR)*, 2015.

[6] Tat-Jun Chin, Pulak Purkait, Anders Eriksson, and David Suter. Efficient globally optimal consensus maximisation with tree search. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 39(4):758–772, 2017.

[7] Ondrej Chum and Jiri Matas. Matching with prosac-progressive sample consensus. In *Computer Vision and Pattern Recognition (CVPR)*, 2005.

[8] Ondřej Chum, Jiří Matas, and Josef Kittler. Locally optimized RANSAC. In *Joint Pattern Recognition Symposium*, 2003.

[9] Olof Enqvist, Erik Ask, Fredrik Kahl, and Kalle Åström. Robust fitting for multiple view geometry. In *European Conference on Computer Vision (ECCV)*, 2012.

[10] David Eppstein. Quasiconvex programming. *Combinatorial and Computational Geometry*, 52(3):287–331, 2005.

[11] Martin A. Fischler and Robert C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[12] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Computer Vision and Pattern Recognition (CVPR)*, 2012.

[13] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2003.

[14] Huu Le, Tat-Jun Chin, and David Suter. An exact penalty method for locally convergent maximum consensus. In *Computer Vision and Pattern Recognition (CVPR)*, 2017.

[15] Hongdong Li. A practical algorithm for L_∞ triangulation with outliers. In *Computer Vision and Pattern Recognition (CVPR)*, 2007.

[16] Hongdong Li. Consensus set maximization with guaranteed global optimality for robust geometry estimation. In *International Conference on Computer Vision (ICCV)*, 2009.

[17] David G Lowe. Object recognition from local scale-invariant features. In *International Conference on Computer Vision (ICCV)*, 1999.

[18] Jiří. Matoušek. On geometric optimization with few violated constraints. *Discrete and Computational Geometry*, 14(4):365–384, 1995.

[19] Jorge Nocedal and Stephen Wright. *Numerical Optimization*. Springer Science & Business Media, 2006.

[20] Carl Olsson, Olof Enqvist, and Fredrik Kahl. A polynomial-time bound for matching and registration with outliers. In *Computer Vision and Pattern Recognition (CVPR)*, 2008.

[21] Carl Olsson, Anders P Eriksson, and Fredrik Kahl. Efficient optimization for L_∞-problems using pseudoconvexity. In *International Conference on Computer Vision (ICCV)*, 2007.

[22] Alvaro Parra Bustos and Tat-Jun Chin. Guaranteed outlier removal for rotation search. In *International Conference on Computer Vision (ICCV)*, 2015.

[23] Pulak Purkait, Christopher Zach, and Anders Eriksson. Maximum consensus parameter estimation by reweighted L1 methods. In *Energy Minimization Methods in Computer Vision and Pattern Recognition (EMMCVPR)*, 2017.

[24] Rahul Raguram, Ondrej Chum, Marc Pollefeys, Jiri Matas, and Jan-Michael Frahm. USAC: a universal framework for random sample consensus. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 35(8):2022–2038, 2013.

[25] Micha Sharir and Emo Welzl. A combinatorial bound for linear programming and related problems. In *Annual Symposium on Theoretical Aspects of Computer Science*, 1992.

[26] Ben J Tordoff and David W Murray. Guided-MLESAC: Faster image transform estimation by using matching priors. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 27(10):1523–1535, 2005.

[27] Andrea Vedaldi and Brian Fulkerson. VLFeat: An open and portable library of computer vision algorithms. In *ACM International Conference on Multimedia*, 2010.

[28] Yinqiang Zheng, Shigeki Sugimoto, and Masatoshi Okutomi. Deterministically maximizing feasible subsystems for robust model fitting with unit norm constraints. In *Computer Vision and Pattern Recognition (CVPR)*, 2011.

Chapter 6

Practical Optimal Registration of Terrestrial LiDAR Scan Pairs

The work contained in this chapter has been published as the following papers

Zhipeng Cai, Tat-Jun Chin, Alvaro Parra Bustos, Konrad Schindler: Practical Optimal Registration of Terrestrial LiDAR Scan Pairs. ISPRS Journal of Photogrammetry and Remote Sensing 2019.

Statement of Authorship

Title of Paper	Practical Optimal Registration of Terrestrial LiDAR Scan Pairs
Publication Status	☑ Published ☐ Accepted for Publication ☐ Submitted for Publication ☐ Unpublished and Unsubmitted work written in manuscript style
Publication Details	Cai, Zhipeng, Tat-Jun Chin, Alvaro Parra Bustos, and Konrad Schindler. "Practical optimal registration of terrestrial LiDAR scan pairs." ISPRS journal of photogrammetry and remote sensing 147 (2019): 118-131.

Principal Author

Name of Principal Author (Candidate)	Zhipeng Cai		
Contribution to the Paper	Proposing the main idea. Conducting experiments. Paper writing.		
Overall percentage (%)	55%		
Certification:	This paper reports on original research I conducted during the period of my Higher Degree by Research candidature and is not subject to any obligations or contractual agreements with a third party that would constrain its inclusion in this thesis. I am the primary author of this paper.		
Signature		Date	16/2/2020

Co-Author Contributions

By signing the Statement of Authorship, each author certifies that:

i. the candidate's stated contribution to the publication is accurate (as detailed above);

ii. permission is granted for the candidate in include the publication in the thesis; and

iii. the sum of all co-author contributions is equal to 100% less the candidate's stated contribution.

Name of Co-Author	Tat-Jun Chin		
Contribution to the Paper	Providing major discussions and suggestions about the method and experiments. Modifications of the paper draft.		
Signature		Date	15/2/2020

Name of Co-Author	Alvaro Parra Bustos		
Contribution to the Paper	Providing suggestions and important source code during the experiment.		
Signature		Date	16/2/2020

Name of Co-Author	Konrad Schindler			
Contribution to the Paper	Providing discussions and suggestions about the method and experiments. Modifications of the paper draft.			
Signature			Date	17/02/2020

Practical optimal registration of terrestrial LiDAR scan pairs

Zhipeng Cai[a,*], Tat-Jun Chin[a], Alvaro Parra Bustos[a], Konrad Schindler[b]

[a]*School of Computer Science, The University of Adelaide, Australia.*
[b]*Institute of Geodesy and Photogrammetry, ETH Zurich, Switzerland.*

Abstract

Point cloud registration is a fundamental problem in 3D scanning. In this paper, we address the frequent special case of registering terrestrial LiDAR scans (or, more generally, levelled point clouds). Many current solutions still rely on the Iterative Closest Point (ICP) method or other heuristic procedures, which require good initializations to succeed and/or provide no guarantees of success. On the other hand, exact or optimal registration algorithms can compute the best possible solution without requiring initializations; however, they are currently too slow to be practical in realistic applications.

Existing optimal approaches ignore the fact that in routine use the relative rotations between scans are constrained to the azimuth, via the built-in level compensation in LiDAR scanners. We propose a novel, optimal and computationally efficient registration method for this 4DOF scenario. Our approach operates on candidate 3D keypoint correspondences, and contains two main steps: (1) a deterministic selection scheme that significantly reduces the candidate correspondence set in a way that is guaranteed to preserve the optimal solution; and (2) a fast branch-and-bound (BnB) algorithm with a novel polynomial-time subroutine for 1D rotation search, that quickly finds the optimal alignment for the reduced set. We demonstrate the practicality of our method on realistic point clouds from multiple LiDAR surveys.

Keywords: Point cloud registration, exact optimization, branch-and-bound.

1. Introduction

LiDAR scanners are a standard instrument in contemporary surveying practice. An individual scan produces a 3D point cloud, consisting of densely sampled, polar line-of-sight measurements of the instrument's surroundings, up to some maximum range. Consequently, a recurrent basic task in LiDAR surveying is to register individual scans into one big point cloud that covers the entire region of interest. The fundamental operation is the relative alignment of a pair of scans. Once this can be done reliably, it can be applied sequentially until all scans are registered; normally followed by a simultaneous refinement of all registration parameters.

Our work focuses on the pairwise registration: given two point clouds, compute the rigid transformation that brings them into alignment. Arguably the most widely used method for this now classical problem is the ICP (Iterative Closest Point) algorithm (Besl and McKay, 1992; Rusinkiewicz and Levoy, 2001; Pomerleau et al., 2013), which alternates between finding nearest-neighbour point matches and updating the transformation parameters. Since the procedure converges only to a local optimum, it requires a reasonably good initial registration to produce correct results.

*Corresponding author
Email addresses: zhipeng.cai@adelaide.edu.au (Zhipeng Cai), tat-jun.chin@adelaide.edu.au (Tat-Jun Chin), alvaro.parrabustos@adelaide.edu.au (Alvaro Parra Bustos), schindler@ethz.ch (Konrad Schindler)

Existing industrial solutions, which are shipped either as on-board software of the scanner itself, or as part of the manufacturer's offline processing software (e.g., Zoller + Fröhlich[1], Riegl[2], Leica[3]), rely on external aids or additional sensors. For instance, a GNSS/IMU sensor package, or a visual-inertial odometry system with a panoramic camera (Houshiar et al., 2015) setup, to enable scan registration in GNSS-denied environments, in particular indoors and under ground. Another alternative is to determine the rotation with a compass, then perform only a translation search, which often succeeds from a rough initialisation, such as setting the translation to 0. Another, older but still popular approach is to install artificial targets in the environment (Akca, 2003; Franaszek et al., 2009) that act as easily detectable and matchable "beacons". However, this comes at the cost of installing and maintaining the targets.

More sophisticated point cloud registration techniques have been proposed that are not as strongly dependent on good initializations, e.g., (Chen et al., 1999; Drost et al., 2010; Albarelli et al., 2010; Theiler et al., 2014, 2015). These techniques, in particular the optimization routines they employ, are heuristic. They often succeed, but cannot guarantee to find an optimal alignment (even according to their own definition of optimality). Moreover, such methods typically are fairly sensitive to the tuning of somewhat unintuitive, input-specific parameters, such as the approximate proportion of overlapping points in 4PCS (Aiger et al., 2008), or the annealing rate of the penalty component in the lifting method of (Zhou et al., 2016). In our experience, when applied to new, unseen registration tasks these methods often do not reach the performance reported on popular benchmark datasets[4].

In contrast to the locally convergent algorithms and heuristics above, optimal algorithms have been developed for point cloud registration (Breuel, 2001; Yang et al., 2016; Campbell and Petersson, 2016; Parra Bustos et al., 2016). Their common theme is to set up a clear-cut, transparent objective function and then apply a suitable exact optimization scheme – often branch-and-bound type methods – to find the solution that maximises the objective function[5]. It is thus ensured that the best registration parameters (according to the adopted objective function) will always be found. Importantly, convergence to the optimal value independent of the starting point implies that these methods do not require initialization. However, a serious limitation of optimal methods is that they are computationally much costlier (due to NP-hardness of most of the robust objective functions used in point cloud registration (Chin et al., 2018)), but also by the experiments in previous literatures, e.g., more than 4 h for only ≈ 350 points in (Parra Bustos et al., 2016), and even longer in (Yang et al., 2016). This makes them impractical for LiDAR surveying.

1.1. Our contributions

In this work, we aim to make optimal registration practical for terrestrial LiDAR scans.

Towards that goal we make the following observations:

- Modern LiDAR devices are equipped with a highly accurate[6] level compensator, which reduces the relative rotation between scans to the azimuth; see Figure 1. In most applications the search space therefore has only 4 degrees of freedom (DOF) rather than 6. This difference is significant, because the runtime of optimal methods

[1] `https://www.zf-laser.com/Z-F-LaserControl-R.laserscanner_software_1.0.html?&L=1`

[2] `http://www.riegl.com/products/software-packages/risolve/`

[3] `https://leica-geosystems.com/products/laser-scanners/software/leica-cyclone`

[4] For example, the Stanford 3D Scanning Repository (Turk and Levoy, 1994).

[5] As opposed to approximate, sub-optimal, or locally optimal solutions with lesser objective values than the maximum achievable.

[6] The tilt accuracy is 0.002° (see page 2 of `http://www.gb-geodezie.cz/wp-content/uploads/2016/01/datenblatt_imager_5006i.pdf`) for the Zoller&Fröhlich Imager 5006i used to capture datasets in our experiment. Similar accuracy can be found in scanners from other companies, e.g., 7.275e-6 radians for the Leica P40 tilt compensator (see page 8 of `https://www.abtech.cc/wp-content/uploads/2017/04/Tilt_compensation_for_Laser_Scanners_WP.pdf`).

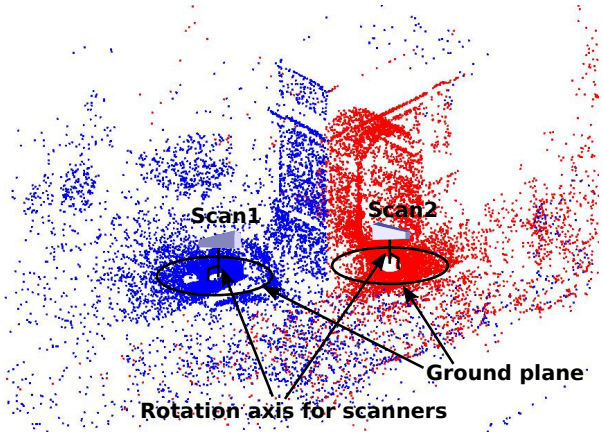


Figure 1: 4DOF registration of two LiDAR scans (blue and red). The level compensator forces scanners to rotate around the vertical axis, resulting in the azimuthal relative rotation.

grows very quickly with the problem dimension, see Figure 2.

- A small set of correspondences, i.e., *correct* point matches referring to close enough locations in the scene, is sufficient to reliably estimate the relative sensor pose, see Figure 2. The problem of match-based registration methods is normally not that there are too few correspondences to solve the problem; but rather that they are drowned in a large number of incorrect point matches, because of the high failure rate of existing 3D matching methods. The task would be a lot easier if we had a way to discard many false correspondences without losing true ones.

On the basis of these observations we develop a novel method for optimal, match-based LiDAR registration, that has the following main steps:

- Instead of operating directly on the input match set, a fast deterministic preprocessing step is executed to aggressively prune the input set, in a way which guarantees that only false correspondences are discarded. In this way, it is ensured that the optimal alignment of the reduced set is the same as for the initial, complete set of matches.
- A fast 4DOF BnB algorithm is then run on the remaining matches to compute the optimal alignment parameters. Our BnB algorithm contains a deterministic polynomial-time subroutine for 1DOF rotation search, which accelerates the optimization.

Figure 2 illustrates our approach. As suggested in the figure and by our comprehensive testing (see Section 6), our approach significantly speeds up optimal registration and makes it practical for realistic LiDAR surveying tasks. For example, on the *Arch*[7] dataset, it is able to accurately register all 5 scans in around 3 min *without* any initializations to the registration; see Figure 3.

Please visit our project homepage[8] for the video demo and source code.

2. Related work

Point-based registration techniques can be broadly categorized into two groups: methods using "raw" point clouds, and methods using 3D point matches. Since our contribution belongs to the second group, we will focus our survey on that group. Nonetheless, in our experiments, we will compare against both raw point cloud methods and match-based methods.

Match-based methods first extract a set of candidate correspondences from the input point clouds (using feature matching techniques such as (Scovanner et al., 2007; Głomb, 2009; Zhong, 2009; Rusu et al., 2008, 2009)), then optimize the registration parameters using the extracted candidates only. Since the accuracy of 3D keypoint detection and matching is much lower than their 2D counterparts (Harris and Stephens, 1988; Lowe, 1999), a major concern of match-based methods is to discount the corrupting effects of false correspondences or outliers.

A widely used strategy for robust estimation is Random Sample Consensus (RANSAC) (Fischler and

[7] http://www.prs.igp.ethz.ch/research/completed_projects/automatic_registration_of_point_clouds.html

[8] https://github.com/ZhipengCai/Demo---Practical-optimal-registration-of-terrestrial-LiDAR-scan-pairs

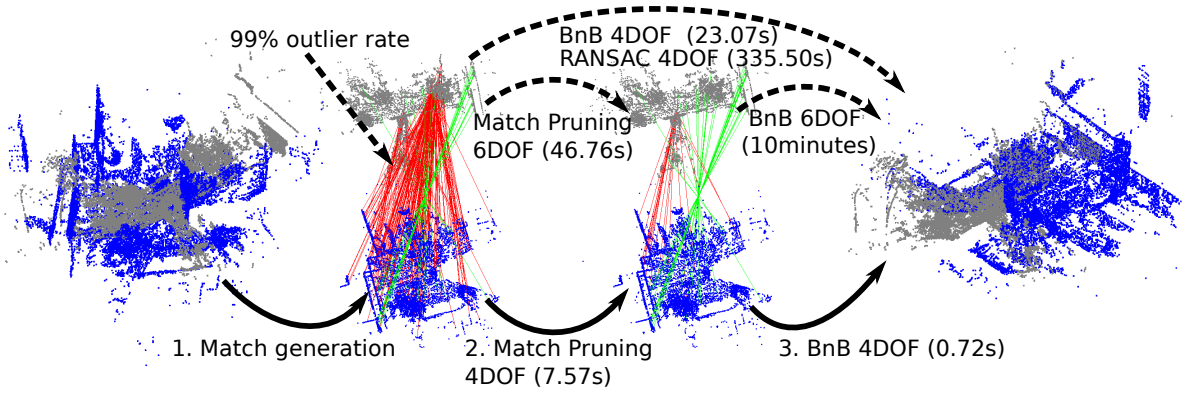


Figure 2: Method illustration. Given 2 point clouds (grey and blue), as shown by the solid arrows, we first generate a set of matches/candidate correspondences (only a subset is shown for visual clarity), which have both inliers (green) and outliers (red). The matches are quickly pruned *without removing any inliers*. Then, we run 4DOF BnB on the remaining matches to find the optimal solution. The dashed arrows show some alternative optimization choices. Note that our two-step process is much faster than directly optimising on the initial matches, and also more practically useful than its 6DOF counterpart (Bustos and Chin, 2017).

Bolles, 1981). However, the runtime of RANSAC increases exponentially with the outlier rate, which is generally very high in 3D registration,e.g., the input match set in Figure 2 contains more than 99% outliers. More efficient approaches have been proposed for dealing with outliers in 3D match sets, such as the game-theoretic method of (Albarelli et al., 2010) and the lifting optimization method of (Zhou et al., 2016). However, these are either heuristic (e.g., using randomisation (Albarelli et al., 2010)) and find solutions that are, at most, *correct with high probability*, but may also be grossly wrong; or they are only locally optimal (Zhou et al., 2016), and will fail when initialized outside of the correct solution's convergence basin.

Optimal algorithms for match-based 3D registration also exist (Bazin et al., 2012; Bustos and Chin, 2017; Yu and Ju, 2018). However, (Bazin et al., 2012) is restricted to pure rotational motion, while the 6DOF algorithms of (Bustos and Chin, 2017; Yu and Ju, 2018) are computationally expensive for unfavourable configurations. E.g., (Bustos and Chin, 2017) takes more than 10 min to register the pair of point clouds in Figure 2.

Recently, clever filtering schemes have been developed (Svarm et al., 2014; Parra Bustos and Chin, 2015; Chin and Suter, 2017; Bustos and Chin, 2017) which have the ability to efficiently prune a set of putative correspondences and only retain a much smaller subset, in a manner that does not affect the optimal solution (more details in Section 5). Our method can be understood as an extension of the 2D rigid registration method of (Chin and Suter, 2017, Section 4.2.1) to the 4DOF case.

Beyond points, other geometric primitives (lines, planes, etc.) have also been exploited for LiDAR registration (Brenner and Dold, 2007; Rabbani et al., 2007). By using more informative order structures in the data, such methods can potentially provide more accurate registration. Developing optimal registration algorithms based on higher-order primitives would be interesting future work.

3. Problem formulation

Given two input point clouds $\mathbf{P}$ and $\mathbf{Q}$, we first extract a set of 3D keypoint matches $\mathcal{C} = \{(\mathbf{p}_i, \mathbf{q}_i)\}_{i=1}^{M}$ between $\mathbf{P}$ and $\mathbf{Q}$. This can be achieved using fairly standard means — Section 6.1 will describe our version of the procedure. Given $\mathcal{C}$, our task is to estimate

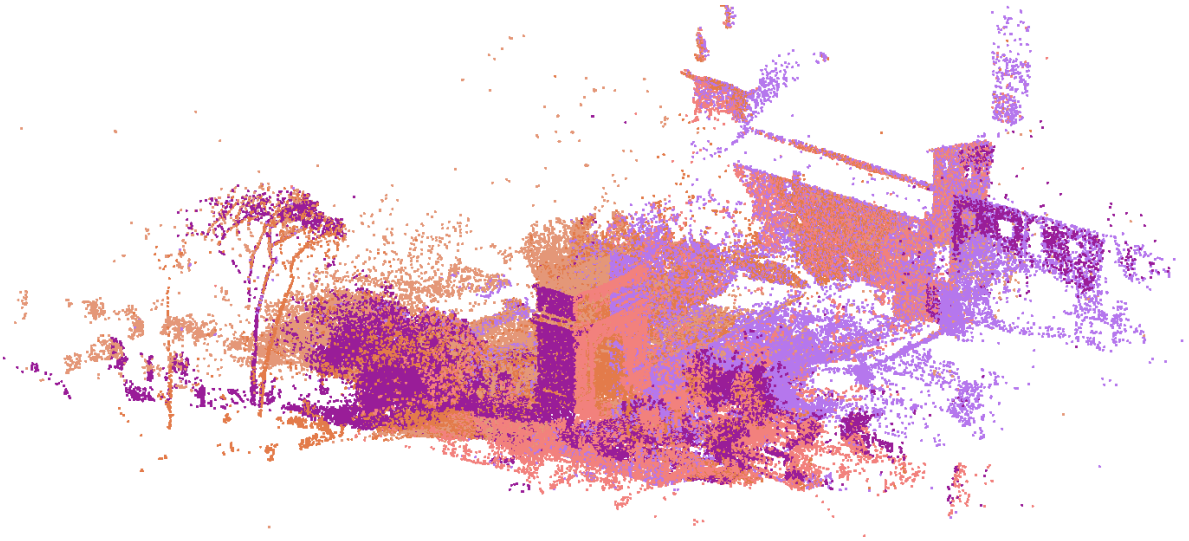


Figure 3: Registration result of our method for *Arch* dataset. The registration of all five scans (with 15k-20k input point matches per pair, not shown) takes 187.53s, without requiring manual initializations. Note that our main contribution in this paper is a fast optimal algorithm for registering LiDAR scan pairs. To register multiple scans (as achieved on the *Arch* dataset in this figure), we sequentially register the individual scans; see Section 3.2 for details.

the 4DOF rigid transformation

$$f(\mathbf{p} \mid \theta, \mathbf{t}) = \mathbf{R}(\theta)\mathbf{p} + \mathbf{t}, \tag{1}$$

parameterized by a rotation angle $\theta \in [0, 2\pi]$ and translation vector $\mathbf{t} \in \mathbb{R}^3$, that aligns as many of the pairs in $\mathcal{C}$ as possible. Note that

$$\mathbf{R}(\theta) = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2}$$

defines a rotation about the 3rd axis, which we assume to be aligned with gravity, expressing the fact that LiDAR scanners in routine use are levelled.[9]

Since $\mathcal{C}$ contains outliers (false correspondences), f must be estimated robustly. To this end, we seek the parameters $\theta, \mathbf{t}$ that maximize the objective

$$E(\theta, \mathbf{t} \mid \mathcal{C}, \epsilon) = \sum_{i=1}^{M} \mathbb{I}\left(\|\mathbf{R}(\theta)\mathbf{p}_i + \mathbf{t} - \mathbf{q}_i\| \leq \epsilon\right), \tag{3}$$

where ϵ is the *inlier threshold*, and $\mathbb{I}$ is an indicator function that returns 1 if the input predicate is satisfied and 0 otherwise. Intuitively, (3) calculates the number of pairs in $\mathcal{C}$ that are aligned up to distance ϵ by $f(\mathbf{p} \mid \theta, \mathbf{t})$. Allowing alignment only up to ϵ is vital to exclude the influence of the outliers. Note that choosing the right ϵ is usually not an obstacle, since LiDAR manufacturers specify the precision of the device, which can inform the choice of an appropriate threshold. Moreover, given the fast performance of the proposed technique, one could conceivably run multiple rounds of registration with different ϵ and choose one based on the alignment residuals.

Our overarching aim is thus to solve the optimization problem

$$E^* = \max_{\theta, \mathbf{t}} E(\theta, \mathbf{t} \mid \mathcal{C}, \epsilon) \tag{4}$$

exactly or optimally; in other words we are searching for the angle and translation vector $\theta^*, \mathbf{t}^*$ that yield the highest objective value $E^* = E(\theta^*, \mathbf{t}^* \mid \mathcal{C}, \epsilon)$. We note that in the context of registration, optimality is not merely an academic exercise. Incorrect local min-

[9] The method is general and will work for any setting that allows only a 1D rotation around a known axis.

Algorithm 1 Main algorithm.

Require: Point clouds $\mathbf{P}$ and $\mathbf{Q}$ with 1D relative rotation, inlier threshold ϵ.

1: Extract match set $\mathcal{C}$ from $\mathbf{P}$ and $\mathbf{Q}$ (Section 6.1).
2: Prune $\mathcal{C}$ into a smaller subset $\mathcal{C}'$ (Section 5).
3: Solve (4) on $\mathcal{C}'$ to obtain registration parameters $\theta^*, \mathbf{t}^*$ (Section 4).
4: **return** Optimal solution $\theta^*, \mathbf{t}^*$.

ima are a real problem, as amply documented in the literature on ICP and illustrated by regular failures of the automatic registration routines in production software.

3.1. Main algorithm

As shown in Algorithm 1, our approach to solve (4) optimally has two main steps: a deterministic pruning step to reduce $\mathcal{C}$ to a much smaller subset $\mathcal{C}'$ while removing only matches that cannot be correct, hence preserving the optimal solution $\theta^*, \mathbf{t}^*$ in $\mathcal{C}'$ (Section 5); and a fast, custom-tailored BnB algorithm to search for $\theta^*, \mathbf{t}^*$ in $\mathcal{C}'$ (Section 4). For better flow of the presentation, we will describe the BnB algorithm first, before the pruning.

3.2. Registering multiple scans

In some applications, registering multiple scans is required. For this purpose, we can first perform pair-wise registration to estimate the relative poses between consecutive scans. These pair-wise poses can then be used to initialize simultaneous multi-scan registration methods like the maximum-likelihood alignment of (Lu and Milios, 1997) and conduct further optimization.

To refrain from further polishing that would obfuscate the contribution made by our robust pair-wise registration, all multi-scan registration results in this paper are generated by sequential pair-wise registration *only*, i.e., starting from the initial scan, incrementally register the next scan to the current one using Algorithm 1. As shown in Figure 3 and later in Section 6.2, our results are already promising even without further refinement.

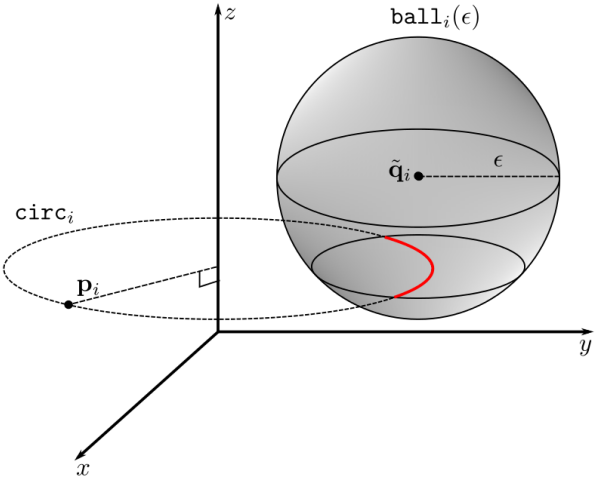


Figure 4: The intersection of $\texttt{circ}_i$ and $\texttt{ball}_i(\epsilon)$. The intersection part is rendered in red.

4. Fast BnB algorithm for 4DOF registration

To derive our fast BnB algorithm, we first rewrite (4) as

$$E^* = \max_{\mathbf{t}} U(\mathbf{t} \mid \mathcal{C}, \epsilon), \tag{5}$$

where

$$U(\mathbf{t} \mid \mathcal{C}, \epsilon) = \max_{\theta} E(\theta, \mathbf{t} \mid \mathcal{C}, \epsilon). \tag{6}$$

It is not hard to see the equivalence of (4) and (5). The purpose of (5) is twofold:

- As we will see in Section (4.1), estimating θ *given* $\mathbf{t}$ can be accomplished deterministically in polynomial time, such that the optimization of θ can be viewed as "evaluating" the function $U(\mathbf{t} \mid \mathcal{C}, \epsilon)$.

- By "abstracting away" the variable θ as in (5), the proposed BnB algorithm (Section 4.2) can be more conveniently formulated as searching over only the 3-dimensional translation space $\mathbb{R}^3$.

4.1. Deterministic rotation estimation

For completeness, the definition of $U(\mathbf{t} \mid \mathcal{C}, \epsilon)$ is as follows

$$U(\mathbf{t} \mid \mathcal{C}, \epsilon) = \max_{\theta} \sum_{i=1}^{M} \mathbb{I}\left(\|\mathbf{R}(\theta)\mathbf{p}_i - \tilde{\mathbf{q}}_i\| \leq \epsilon\right), \quad (7)$$

where $\tilde{\mathbf{q}}_i = \mathbf{q}_i - \mathbf{t}$. Intuitively, evaluating this function amounts to finding the rotation $\mathbf{R}(\theta)$ that aligns as many pairs $\{(\mathbf{p}_i, \tilde{\mathbf{q}}_i)\}_{i=1}^{M}$ as possible.

Note that for each $\mathbf{p}_i$, rotating it with $\mathbf{R}(\theta)$ for all $\theta \in [0, 2\pi]$ forms the circular trajectory

$$\texttt{circ}_i = \{\mathbf{R}(\theta)\mathbf{p}_i \mid \theta \in [0, 2\pi]\}. \quad (8)$$

Naturally, $\texttt{circ}_i$ collapses to a point if $\mathbf{p}_i$ lies on the z-axis. Define

$$\texttt{ball}_i(\epsilon) = \left\{\mathbf{q} \in \mathbb{R}^3 \mid \|\mathbf{q} - \tilde{\mathbf{q}}_i\| \leq \epsilon\right\} \quad (9)$$

as the ϵ-ball centered at $\tilde{\mathbf{q}}_i$. It is clear that the pair $(\mathbf{p}_i, \tilde{\mathbf{q}}_i)$ can only be aligned by $\mathbf{R}(\theta)$ if $\texttt{circ}_i$ and $\texttt{ball}_i(\epsilon)$ intersect; see Figure 4. Moreover, if $\texttt{circ}_i$ and $\texttt{ball}_i(\epsilon)$ do not intersect, then we can be sure that the i-th pair plays no role in (7).

For each i, denote

$$\texttt{int}_i = [\alpha_i, \beta_i] \subseteq [0, 2\pi] \quad (10)$$

as the angular interval such that, for all $\theta \in \texttt{int}_i$, $\mathbf{R}(\theta)\mathbf{p}_i$ is aligned with $\tilde{\mathbf{q}}_i$ within distance ϵ. The limits α_i and β_i can be calculated in closed form via circle-to-circle intersections; see Appendix 9.1. Note that $\texttt{int}_i$ is empty if $\texttt{circ}_i$ and $\texttt{ball}_i(\epsilon)$ do not intersect. For brevity, in the following we take all $\texttt{int}_i$ to be single intervals. For the actual implementation, it is straight-forward to break $\texttt{int}_i$ into two intervals if it extends beyond the range $[0, 2\pi]$. The function $U(\mathbf{t} \mid \mathcal{C}, \epsilon)$ can then be rewritten as

$$U(\mathbf{t} \mid \mathcal{C}, \epsilon) = \max_{\theta} \sum_{i=1}^{M} \mathbb{I}\left(\theta \in [\alpha_i, \beta_i]\right), \quad (11)$$

which is an instance of the *max-stabbing* problem (De Berg et al., 2000, Chapter 10); see Figure 5. Efficient algorithms for max-stabbing are known, in particular, the version in Algorithm 5 in the Appendix runs deterministically in $\mathcal{O}(M \log M)$ time. This supports a practical optimal algorithm.

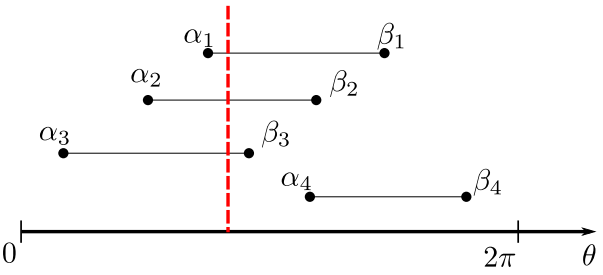


Figure 5: The max-stabbing problem aims to find a vertical line (defined by angle θ in our case) that "stabs" the maximum number of intervals, e.g., the dashed red line. Note that 0 and 2π refer to the same 1D rotation. To ensure their stabbing values are equal, if an interval has only one end on one of these two angles, an extra interval that starts and ends both at the other angle is added to the input.

4.2. BnB for translation search

In the context of solving (5), the BnB method initializes a cube $\mathbb{S}_0$ in $\mathbb{R}^3$ that contains the optimal solution $\mathbf{t}^*$, then recursively partitions $\mathbb{S}_0$ into 8 sub-cubes; see Algorithm 2. For each sub-cube $\mathbb{S} \subset \mathbb{S}_0$, let $\mathbf{t}_{\mathbb{S}}$ be the center point of $\mathbb{S}$. If $\mathbf{t}_{\mathbb{S}}$ gives a higher objective value than the current best estimate $\hat{\mathbf{t}}$, the latter is updated to become the former, $\hat{\mathbf{t}} \leftarrow \mathbf{t}_{\mathbb{S}}$; else, either

- a decision is made to discard $\mathbb{S}$ (see below); or
- $\mathbb{S}$ is partitioned into 8 sub-cubes and the process above is repeated.

In the limit, $\hat{\mathbf{t}}$ approaches the optimal solution $\mathbf{t}^*$.

Given a sub-cube $\mathbb{S}$ and the incumbent solution $\hat{\mathbf{t}}$, $\mathbb{S}$ is discarded if

$$\bar{U}(\mathbb{S} \mid \mathcal{C}, \epsilon) \leq U(\hat{\mathbf{t}} \mid \mathcal{C}, \epsilon), \quad (12)$$

where $\bar{U}(\mathbb{S} \mid \mathcal{C}, \epsilon)$ calculates an *upper bound* of $U(\mathbf{t} \mid \mathcal{C}, \epsilon)$ over domain $\mathbb{S}$, i.e.,

$$\bar{U}(\mathbb{S} \mid \mathcal{C}, \epsilon) \geq \max_{\mathbf{t} \in \mathbb{S}} U(\mathbf{t} \mid \mathcal{C}, \epsilon). \quad (13)$$

The rationale is that if (12) holds, then a solution that is better than $\hat{\mathbf{t}}$ cannot exist in $\mathbb{S}$. In our work, the upper bound is obtained as

$$\bar{U}(\mathbb{S} \mid \mathcal{C}, \epsilon) = U(\mathbf{t}_{\mathbb{S}} \mid \mathcal{C}, \epsilon + d_{\mathbb{S}}), \quad (14)$$

Algorithm 2 BnB for 4DOF match-based registration (5).

Require: Initial matches $\mathcal{C}$ and the inlier threshold ϵ.

1: Set the priority queue w to $\emptyset$, $\mathbb{S}_0 \leftarrow$ the initial translation cube, $\hat{\mathbf{t}} \leftarrow \mathbf{t}_{\mathbb{S}_0}$.
2: Compute $\bar{U}(\mathbb{S}_0 \mid \mathcal{C}, \epsilon)$ by Algorithm 5 and insert $(\mathbb{S}_0, \bar{U}(\mathbb{S}_0 \mid \mathcal{C}, \epsilon))$ into w.
3: **while** w is not empty **do**
4: Pop out the cube $\mathbb{S}$ with the highest $\bar{U}(\mathbb{S} \mid \mathcal{C}, \epsilon)$ from w.
5: Compute $U(\mathbf{t}_{\mathbb{S}} \mid \mathcal{C}, \epsilon)$ by Algorithm 5; If $U(\mathbf{t}_{\mathbb{S}} \mid \mathcal{C}, \epsilon) = \bar{U}(\mathbb{S} \mid \mathcal{C}, \epsilon)$, break.
6: If $U(\mathbf{t}_{\mathbb{S}} \mid \mathcal{C}, \epsilon) > U(\hat{\mathbf{t}} \mid \mathcal{C}, \epsilon)$, $\hat{\mathbf{t}} \leftarrow \mathbf{t}_{\mathbb{S}}$ and prune w according to $U(\hat{\mathbf{t}} \mid \mathcal{C}, \epsilon)$.
7: Divide $\mathbb{S}$ into 8 sub-cubes $\{\mathbb{S}_o\}_{o=1}^{8}$ and compute $\bar{U}(\mathbb{S}_o \mid \mathcal{C}, \epsilon)$ by Algorithm 5 for all $\mathbb{S}_o$.
8: For each $\mathbb{S}_o$, if $\bar{U}(\mathbb{S}_o \mid \mathcal{C}, \epsilon) > U(\hat{\mathbf{t}} \mid \mathcal{C}, \epsilon)$, insert $(\mathbb{S}_o, \bar{U}(\mathbb{S}_o \mid \mathcal{C}, \epsilon))$ into w.
9: **end while**
10: **return** $\hat{\mathbf{t}}$.

where $d_{\mathbb{S}}$ is half of the diagonal length of $\mathbb{S}$. Note that computing the bound amounts to evaluating the function U, which can be done efficiently via max-stabbing.

The following lemma establishes the validity of (14) for BnB.

Lemma 1. *For any translation cube* $\mathbb{S} \subset \mathbb{R}^3$,

$$\bar{U}(\mathbb{S} \mid \mathcal{C}, \epsilon) = U(\mathbf{t}_{\mathbb{S}} \mid \mathcal{C}, \epsilon + d_{\mathbb{S}}) \geq \max_{\mathbf{t} \in \mathbb{S}} U(\mathbf{t} \mid \mathcal{C}, \epsilon), \tag{15}$$

and as $\mathbb{S}$ *tends to a point* $\mathbf{t}$*, then*

$$\bar{U}(\mathbb{S} \mid \mathcal{C}, \epsilon) \to U(\mathbf{t} \mid \mathcal{C}, \epsilon). \tag{16}$$

See Appendix 9.2 for the proofs.

To conduct the search more strategically, in Algorithm 2 the unexplored sub-cubes are arranged in a priority queue w based on their upper bound value. Note that while Algorithm 2 appears to be solving only for the translation $\mathbf{t}$ via problem (5), *implicitly* it is simultaneously optimizing the angle θ: given the output $\mathbf{t}^*$ from Algorithm 2, the optimal θ^* per the original problem (4) can be obtained by evaluating $U(\mathbf{t}^* \mid \mathcal{C}, \epsilon)$ and keeping the maximizer.

5. Fast preprocessing algorithm

Instead of invoking BnB (Algorithm 2) on the match set $\mathcal{C}$ directly, our approach first executes a preprocessing step (see Step 2 in Algorithm 1) to reduce $\mathcal{C}$ to a much smaller subset $\mathcal{C}'$, then runs BnB on $\mathcal{C}'$. Remarkably. this pruning can be carried out in such a way that the optimal solution is preserved in $\mathcal{C}'$, i.e.,

$$\theta^*, \mathbf{t}^* = \arg\max_{\theta, \mathbf{t}} \; E(\theta, \mathbf{t} \mid \mathcal{C}, \epsilon) = \arg\max_{\theta, \mathbf{t}} \; E(\theta, \mathbf{t} \mid \mathcal{C}', \epsilon). \tag{17}$$

Hence, BnB runs a lot faster, but still finds the optimum w.r.t. the original, full match set.

Let $\mathcal{I}^*$ be the subset of $\mathcal{C}$ that are aligned by $\theta^*, \mathbf{t}^*$, formally

$$\|\mathbf{R}(\theta^*)\mathbf{p}_i + \mathbf{t}^* - \mathbf{q}_i\| \leq \epsilon \;\; \forall (\mathbf{p}_i, \mathbf{q}_i) \in \mathcal{I}^*. \tag{18}$$

If the following condition holds

$$\mathcal{I}^* \subseteq \mathcal{C}' \subseteq \mathcal{C}, \tag{19}$$

then it follows that (17) will also hold. Thus, the trick for preprocessing is to remove only matches that are in $\mathcal{C} \setminus \mathcal{I}^*$, i.e., "certain outliers".

5.1. Identifying the certain outliers

To accomplish the above, define the problem $\mathcal{P}[k]$[10]:

$$\begin{aligned} \max_{\theta, \mathbf{t}} \quad & 1 + \sum_{i \in \mathcal{J}_k} \mathbb{I}\left(\|\mathbf{R}(\theta)\mathbf{p}_i + \mathbf{t} - \mathbf{q}_i\| \leq \epsilon\right) \\ \text{s.t.} \quad & \|\mathbf{R}(\theta)\mathbf{p}_k + \mathbf{t} - \mathbf{q}_k\| \leq \epsilon, \end{aligned} \tag{20}$$

where $k \in \{1, \ldots, M\}$, and $\mathcal{J}_k = \{1, \ldots, M\} \setminus \{k\}$. In words, $\mathcal{P}[k]$ is the same problem as the original

[10]Note that the "1+" in (20) is necessary because we want the optimal value E_k^* of $\mathcal{P}[k]$ to be exactly equal to E^* if the k-th match is an inlier, which is the basis of Lemma 2.

registration problem (4), except that the k-th match must be aligned. We furthermore define E_k^* as the optimal value of $\mathcal{P}[k]$; $\bar{E}_k$ as an upper bound on the value of $\mathcal{P}[k]$, such that $\bar{E}_k \geq E_k^*$; and $\underline{E}$ as a lower bound on the value of (4), so $\underline{E} \leq E^*$. Note that, similar to E^*, E_k^* can only be obtained by optimal search using BnB, but we want to avoid such a costly computation in the pruning stage. Instead, if we have access to $\bar{E}_k$ and $\underline{E}$ (details in Section 5.2), the following test can be made.

Lemma 2. *If $\bar{E}_k < \underline{E}$, then $(\mathbf{p}_k, \mathbf{q}_k)$ is a certain outlier.*

Proof. If $(\mathbf{p}_k, \mathbf{q}_k)$ is in $\mathcal{I}^*$, then we must have that $E_k^* = E^*$, i.e., $\mathcal{P}[k]$ and (4) must have the same solution. However, if we are given that $\bar{E}_k < \underline{E}$, then

$$E_k^* \leq \bar{E}_k < \underline{E} \leq E^* \tag{21}$$

which contradicts the previous statement. Thus, $(\mathbf{p}_k, \mathbf{q}_k)$ cannot be in $\mathcal{I}^*$. □

The above lemma underpins a pruning algorithm that removes certain outliers from $\mathcal{C}$ in order to reduce it to a smaller subset $\mathcal{C}'$, which still includes all inlier putative correspondences; see Algorithm 3. The algorithm simply iterates over $k = 1, \ldots, M$ and attempts the test in Lemma 2 to remove $(\mathbf{p}_k, \mathbf{q}_k)$. At each k, the upper and lower bound values $\bar{E}_k$ and $\underline{E}$ are computed and/or refined (details in the following section). As we will demonstrate in Section 6, Algorithm 3 is able to shrink $\mathcal{C}$ to less than 20% of its original size for practical cases.

5.2. Efficient bound computation

For the data in problem $\mathcal{P}[k]$, let them be centered w.r.t. $\mathbf{p}_k$ and $\mathbf{q}_k$, i.e.,

$$\mathbf{p}_i' = \mathbf{p}_i - \mathbf{p}_k, \quad \mathbf{q}_i' = \mathbf{q}_i - \mathbf{q}_k, \quad \forall i. \tag{22}$$

Then, define the following pure rotational problem $\mathcal{Q}[k]$:

$$\max_{\theta} \quad 1 + \sum_{i \in \mathcal{J}_k} \mathbb{I}\left(\|\mathbf{R}(\theta)\mathbf{p}_i' - \mathbf{q}_i'\| \leq 2\epsilon\right). \tag{23}$$

Algorithm 3 Fast match pruning (FMP) for 4DOF registration.

Require: Initial matches $\mathcal{C}$, the inlier threshold ϵ.
1: $\underline{E} \leftarrow 0$, $\mathcal{C}' \leftarrow \mathcal{C}$.
2: **for** $k = 1, ..., M$ **do**
3: Compute $\bar{E}_k$ (Section 5.2).
4: **if** $\bar{E}_k < \underline{E}$ **then**
5: $\mathcal{C}' \leftarrow \mathcal{C}' \backslash (\mathbf{p}_k, \mathbf{q}_k)$.
6: **else**
7: Re-evaluate $\underline{E}$ using the corresponding solution of $\bar{E}_k$ (Section 5.2).
8: **end if**
9: **end for**
10: Remove from $\mathcal{C}'$ the remaining $(\mathbf{p}_k, \mathbf{q}_k)$ whose $\bar{E}_k < \underline{E}$.
11: **return** $\mathcal{C}'$

We now show that $\bar{E}_k$ and $\underline{E}$ in Algorithm 3 can be computed by solving $\mathcal{Q}[k]$, which can again be done efficiently using max-stabbing (Algorithm 5).

First, we show by Lemma 3 that the value of $\mathcal{Q}[k]$ can be directly used as $\bar{E}_k$, i.e., the number of inliers in $\mathcal{Q}[k]$ is an upper bound of the one in $\mathcal{P}[k]$.

Lemma 3. *If $(\mathbf{p}_i, \mathbf{q}_i)$ is aligned by the optimal solution θ_k^* and $\mathbf{t}_k^*$ of $\mathcal{P}[k]$, $(\mathbf{p}_i', \mathbf{q}_i')$ must also be aligned by θ_k^* in $\mathcal{Q}[k]$.*

Proof. To align $(\mathbf{p}_k, \mathbf{q}_k)$ in $\mathcal{P}[k]$, $\mathbf{t}_k^*$ must be within the ϵ-ball centered at $\mathbf{q}_k - \mathbf{R}(\theta_k^*)\mathbf{p}_k$, i.e., $\mathbf{t}_k^*$ can be re-expressed by $\mathbf{q}_k - \mathbf{R}(\theta_k^*)\mathbf{p}_k + \mathbf{t}_k^{*\prime}$, where $\|\mathbf{t}_k^{*\prime}\| \leq \epsilon$. Using this re-expression, when $(\mathbf{p}_i, \mathbf{q}_i)$ is aligned by θ_k^* and $\mathbf{t}_k^*$, we have

$$\|\mathbf{R}(\theta_k^*)\mathbf{p}_i + (\mathbf{q}_k - \mathbf{R}(\theta_k^*)\mathbf{p}_k + \mathbf{t}_k^{*\prime}) - \mathbf{q}_i\| \tag{24}$$

$$= \|\mathbf{R}(\theta_k^*)\mathbf{p}_i' + \mathbf{t}_k^{*\prime} - \mathbf{q}_i'\| \leq \epsilon \tag{25}$$

$$\Rightarrow \|\mathbf{R}(\theta_k^*)\mathbf{p}_i' - \mathbf{q}_i'\| - \|\mathbf{t}_k^{*\prime}\| \leq \epsilon \tag{26}$$

$$\Rightarrow \|\mathbf{R}(\theta_k^*)\mathbf{p}_i' - \mathbf{q}_i'\| - \epsilon \leq \epsilon \Leftrightarrow \|\mathbf{R}(\theta_k^*)\mathbf{p}_i' - \mathbf{q}_i'\| \leq 2\epsilon, \tag{27}$$

(26) and (27) are due respectively to the triangle inequality[11] and to $\|\mathbf{t}_k^{*\prime}\| \leq \epsilon$. According to (27), $(\mathbf{p}_i', \mathbf{q}_i')$ is also aligned by θ_k^* in $\mathcal{Q}[k]$. □

[11] https://en.wikipedia.org/wiki/Triangle_inequality

On the other hand, $\underline{E}$, the lower bound of E^*, can be calculated using the optimal solution $\tilde{\theta}_k$ of $\mathcal{Q}[k]$. Specifically, we set $\tilde{\mathbf{t}}_k = \mathbf{q}_k - \mathbf{R}(\tilde{\theta}_k)\mathbf{p}_k$ and compute $\underline{E} = U(\tilde{\mathbf{t}}_k \mid \mathcal{C}', \epsilon) = E(\tilde{\theta}_k, \tilde{\mathbf{t}}_k \mid \mathcal{C}', \epsilon)$, directly following Equation (3).

In this way, evaluating $\bar{E}_k$ and $\underline{E}$ takes $\mathcal{O}(M \log M)$, respectively $\mathcal{O}(M)$ time. As Algorithm 3 repeats both evaluations M times, its time complexity is $\mathcal{O}(M^2 \log M)$.

6. Experiments

The experiments contain two parts, which show respectively the results on *controlled* and *real-world* data. All experiments were implemented in C++ and executed on a laptop with 16 GB RAM and Intel Core 2.60 GHz i7 CPU.

6.1. Controlled data

To refer to our fast match pruning step (Algorithm 3), we abbreviate it as FMP in the rest of this paper. We first show the effect of FMP to the speed of our method, by comparing

- FMP + BnB: Algorithm 2 with FMP for preprocessing;
- BnB: Algorithm 2 without preprocessing;
- FMP: Algorithm 3;

on data with varied overlap ratios τ. Varying τ showed the performance for different outlier rates. And to preserve the effect of feature matching, we chose to manipulate τ instead of the outlier rate directly. τ was controlled by sampling two subsets from a complete point cloud (*Bunny* and *Armadillo*[12] in this experiment). Each subset contains $\lfloor \frac{2}{4-2\tau} \rfloor$ of all points, and the second subset is displaced relative to the first one with a randomly generated 4DOF transformation. (3D translation and rotation around the vertical). Moreover, the point clouds are rescaled to have an average point-to-point distance of 0.05 m, and contaminated with uniform random noise of magnitude $[0 \dots 0.05]$ m.

Given two point clouds, the initial match set $\mathcal{C}$ was generated (here and in all subsequent experiments) by

1. Voxel Grid (Lee et al., 2001) down-sampling and ISS (Zhong, 2009) keypoint extraction;
2. FPFH feature (Rusu et al., 2009) computation and matching on keypoints. $\mathbf{p}_i$ and $\mathbf{q}_i$ were selected into $\mathcal{C}$ if their FPFH features are one of the λ nearest neighbours *to each other*. Empirically, λ needs to be a bit larger than 1 to generate enough inliers. We set $\lambda = 10$ in all experiments.

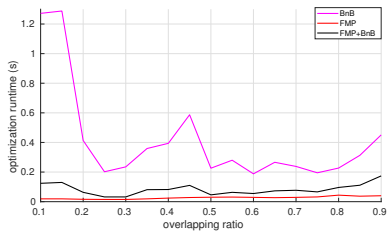
The inlier threshold ϵ was set to 0.05 m according to the noise level.

Figure 6 reports the runtime of the three algorithms and the number of matches before and after FMP, with τ varied from 0.1 to 0.9. See Table 1 for the input size. As shown in Figure 6(a) and 6(b), due to the extremely high outlier rate, BnB was much slower when τ is small, whereas FMP + BnB remained efficient for all τ. This significant acceleration came from the drastically reduced input size (more than 90% of the outliers were pruned) after executing FMP (Figure 6(c) and 6(d)), and the extremely low overhead of FMP.

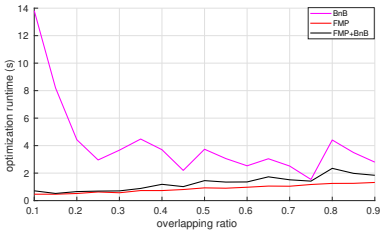
Note that the data storing order of $\mathcal{C}$ was used as the data processing order (the order of k in the for-loop of Algorithm 3) in FMP for all registration tasks in this paper. Though theoretically the data processing order of FMP does affect the size $|\mathcal{C}'|$ of the pruned match set, in practice, this effect is minor. To show the stability of FMP under different data processing orders, we executed FMP 100 times given the same input match set, but with randomly shuffled data processing order in each FMP execution. Figures 6(e) and 6(f) report the median, minimum and maximum value of $|\mathcal{C}'|$ in 100 runs, and the value of $|\mathcal{C}'|$ using the original data storing order as the data processing order. As can be seen, the value $|\mathcal{C}'|$ is stable across all instances.

To show the advantage of our method, we also compared it against the following state-of-the-art approaches, on the same set of data.

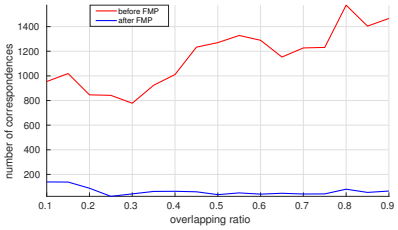
[12] `http://graphics.stanford.edu/data/3Dscanrep/`



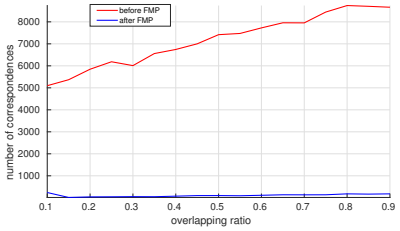
(a) Runtime—*Bunny*.



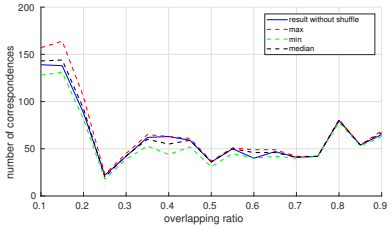
(b) Runtime—*Armadillo*.



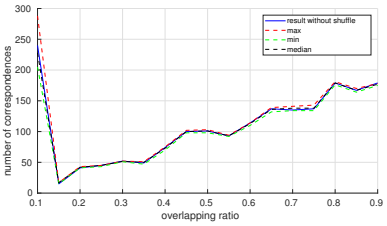
(c) Number of matches before and after FMP—*Bunny*.



(d) Number of matches before and after FMP—*Armadillo*.



(e) Statistics of $|\mathcal{C}'|$ in 100 runs of FMP with shuffled data processing order—*Bunny*.



(f) Statistics of $|\mathcal{C}'|$ in 100 runs of FMP with shuffled data processing order—*Armadillo*.

Figure 6: The effect of FMP on data with varied overlapping ratios τ. The runtime of only FMP was reported to show the portion of runtime in FMP + BnB spending on FMP.

- 4DOF RANSAC (Fischler and Bolles, 1981), using 2-point samples for 4DOF pose estimation,and with the probability of finding a valid sample set to 0.99[13];

- 4DOF version of the lifting method (LM) (Zhou et al., 2016), a match-based robust optimization approach[14]. The annealing rate was tuned to 1.1 for the best performance;

- The Game-Theory approach (GTA) (Albarelli et al., 2010), a fast outlier removal method[15];

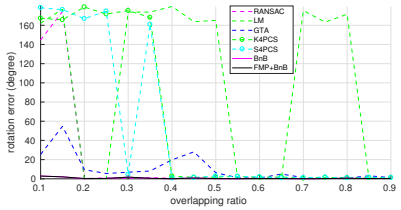
[13] C++ code in `https://bitbucket.org/Zhipeng_Cai/isprsjdemo/src/default/`.

[14] C++ implementation based on the code from `http://vladlen.info/publications/fast-global-registration/`.

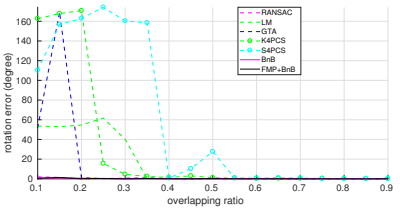
[15] C++ implementation based on the code from `http://www.isi.imi.i.u-tokyo.ac.jp/~rodola/sw.html` and `http://vision.in.tum.de/_media/spezial/bib/cvpr12-code.zip`.

Data	*Bunny*					*Armadillo*				
τ	$\lvert\mathbf{P}\rvert$	$\lvert\mathbf{Q}\rvert$	$\lvert\mathbf{P}_{key}\rvert$	$\lvert\mathbf{Q}_{key}\rvert$	$\lvert\mathcal{C}\rvert$	$\lvert\mathbf{P}\rvert$	$\lvert\mathbf{Q}\rvert$	$\lvert\mathbf{P}_{key}\rvert$	$\lvert\mathbf{Q}_{key}\rvert$	$\lvert\mathcal{C}\rvert$
0.10	15499	15499	233	140	954	77927	77927	1410	820	5097
0.15	15918	15918	248	139	1019	80033	80033	1478	863	5369
0.20	16360	16360	257	147	846	82256	82256	1554	922	5846
0.25	16827	16827	266	153	842	84606	84606	1613	967	6185
0.30	17322	17322	276	155	778	87095	87095	1676	981	6010
0.35	17847	17847	290	160	924	89734	89734	1745	1045	6558
0.40	18405	18405	298	173	1011	92538	92538	1809	1083	6744
0.45	18999	18999	307	170	1233	95524	95524	1862	1136	7000
0.50	19632	19632	321	175	1270	98708	98708	1923	1140	7417
0.55	20309	20309	332	182	1328	102111	102111	1969	1198	7472
0.60	21034	21034	340	190	1289	105758	105758	2033	1216	7725
0.65	21813	21813	348	193	1153	109675	109675	2053	1284	7957
0.70	22652	22652	380	200	1227	113894	113894	2114	1284	7953
0.75	23558	23558	390	205	1231	118449	118449	2160	1360	8444
0.80	24540	24540	400	221	1574	123385	123385	2253	1407	8741
0.85	25607	25607	413	228	1404	128749	128749	2297	1441	8706
0.90	26771	26771	426	227	1466	134602	134602	2352	1475	8664

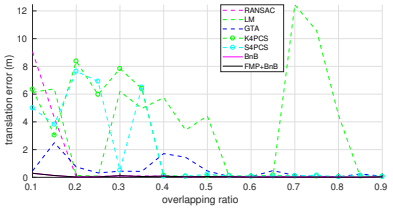
Table 1: Size of the controlled data. $|\mathbf{P}|$ and $|\mathbf{Q}|$: size of the input point clouds. $|\mathbf{P}_{key}|$ and $|\mathbf{Q}_{key}|$: number of keypoints.



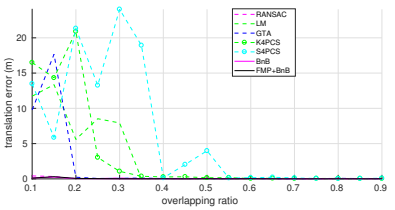
(a) Rotation error—*Bunny*.



(b) Rotation error—*Armadillo*.



(c) Translation error—*Bunny*.



(d) Translation error—*Armadillo*.

Figure 7: The registration accuracy of all methods on data with varied overlapping ratios τ.

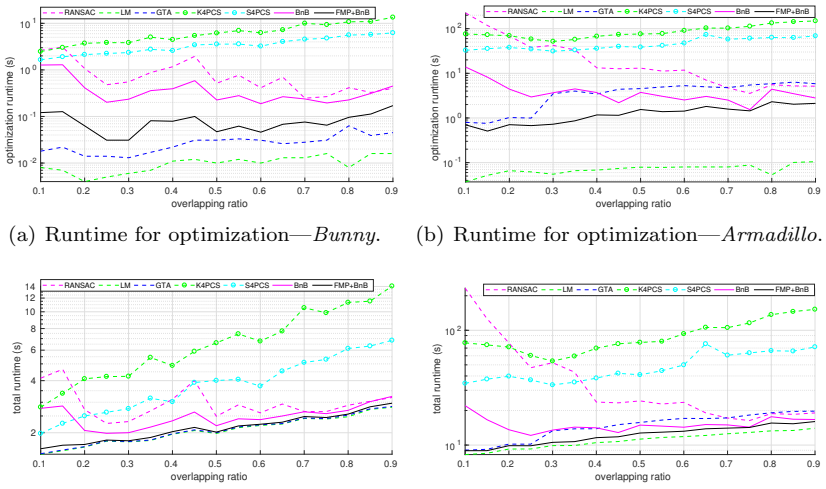
- Super 4PCS (S4PCS) (Mellado et al., 2014), a fast 4PCS (Aiger et al., 2008) variant[16].

- Keypoint-based 4PCS (K4PCS) (Theiler et al., 2014), which applies 4PCS to keypoints[17].

The first three approaches are match-based and the last two operates on raw point sets (ISS keypoints were used here). The 4DOF version of RANSAC and LM were used for a fair comparison, since they had similar accuracy but were much faster than their 6DOF counterparts. We note that, when working with levelled point clouds, the translation alone must already align the z-coordinates of two points up to the inlier threshold ϵ. This can be checked before

[16]C++ code from `http://geometry.cs.ucl.ac.uk/projects/2014/super4PCS/`

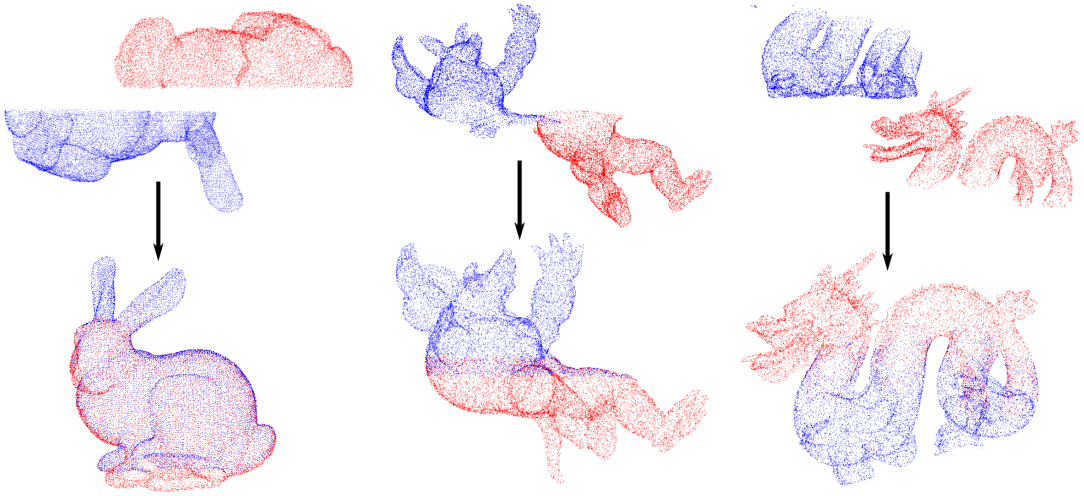
[17]C++ code from PCL: `http://pointclouds.org/`



(a) Runtime for optimization—*Bunny*. (b) Runtime for optimization—*Armadillo*.

(c) Total runtime—*Bunny*. (d) Total runtime—*Armadillo*.

Figure 8: The log scaled runtime of all methods with (up) and without (bottom) input genreration (only generating keypoints for K4PCS and S4PCS) on controlled data.



(a) *Bunny*. (b) *Armadillo*. (c) *Dragon*.

Figure 9: Registered controlled data ($\tau = 0.1$). 10k points are shown for each point cloud.

sampling/applying the rotation. Where applicable, we use this trick to save computations (for all methods). For GTA as well as 4PCS variants, the original versions for 6DOF had to be used, since it is not obvious how to constrain the underlying algorithms to 4DOF.

Arch					
Data	$\|\mathbf{P}\|$	$\|\mathbf{Q}\|$	$\|\mathbf{P}_{key}\|$	$\|\mathbf{Q}_{key}\|$	$\|\mathcal{C}\|$
s01-s02	23561732	30898999	7906	4783	19879
s02-s03	30898999	25249235	4783	7147	19344
s03-s04	25249235	29448437	7147	5337	22213
s04-s05	29448437	27955953	5337	4676	15529
Facility					
Data	$\|\mathbf{P}\|$	$\|\mathbf{Q}\|$	$\|\mathbf{P}_{key}\|$	$\|\mathbf{Q}_{key}\|$	$\|\mathcal{C}\|$
s05-s06	10763194	10675580	2418	2727	10679
s09-s10	10745525	10627814	2960	1327	7037
s12-s13	10711291	10441772	1227	2247	6001
s18-s19	10541274	10602884	1535	2208	7260

Table 2: Size of the real-world data. $|\mathbf{P}_{key}|$ and $|\mathbf{Q}_{key}|$: number of keypoints.

Figure 7 shows the accuracy of all methods, which was measured as the difference between the estimated transformation and the known ground truth. As expected, BnB and FMP + BnB returned high quality solutions for all setups. In contrast, due to the lack of optimality guarantees, other methods performed badly when τ was small. Meanwhile, Figure 8 shows both the runtime including and not including the input generation (keypoint extraction and/or feature matching). FMP + BnB was faster than most of its competitors. Note that most of the total runtime was spent on input generation. Other than sometimes claimed, exact optimization is not necessarily slow and does in fact not create a computational bottleneck for registration. Figure 9 shows some visual examples of point clouds aligned with our method.

6.2. Challenging real-world data

To demonstrate the practicality of our method, comparisons on large sale LiDAR datasets[18] were also performed. Figure 10 reports the accuracy of all methods on an outdoor dataset, *Arch*, and an indoor dataset, *Facility*. Among the datasets used in (Theiler et al., 2015), these are the most challenging ones, which most clearly demonstrate the advantages of our proposed method. We point out that both are not staged, but taken from real surveying projects and representative for the registration problems that arise in difficult field campaigns. For completeness, Appendix 9.3 contains results on easier data, where most of the compared methods work well). The accuracy was again measured by the difference between the estimated and ground truth (provided in the selected datasets, see (Theiler et al., 2015) for details) relative poses. Similar as before, at a reasonable error threshold[19] of 15 cm for translation, respectively 1° for rotation, our method had 100% success rate; whereas failure cases occurred with all other methods. And as shown in Figure 11, FMP + BnB showed comparable speed than its competitors on all data. It successfully registered millions of points and tens of thousands of point matches (see Table 2) within 100 s, including the match generation time. We see the excellent balance between high accuracy and reliability on the one hand, and low computation time on the other hand as a particular strength of our method.

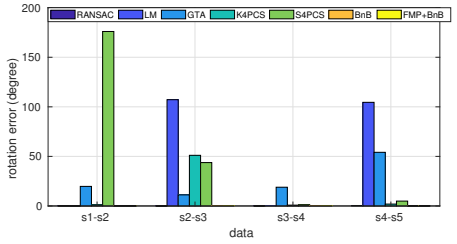
Figures 12–15 visually show various large scale scenes (pair-wise and complete) registered by our method; more detailed demonstrations can be found in the video demo in our project homepage. Note that the runtime for registering a complete dataset (in Figure 3 and 17 to 19) was slightly less than the sum of pair-wise runtimes, since a scan forms part of multiple pairs, but keypoints and FPFH features need to be extracted only once and can then be reused.

7. Conclusion

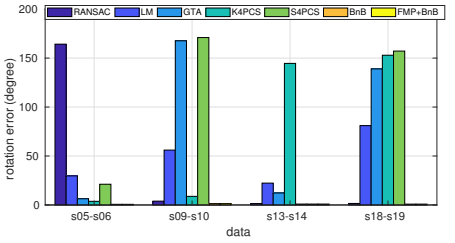
We have described a practical and optimal solution for terrestrial LiDAR scan registration. The main characteristic of our approach is that it combines a reliable, optimal solver with high computational efficiency, by exploiting two properties of terrestrial LiDAR: (1) it restricts the registration problem to 4DOF (3D translation + azimuth), since modern laser scanners have built-in level compensators. And (2) it aggressively prunes the number of matches used to compute the transformation, realising that the sensor noise level is low, therefore a small set of corresponding points is sufficient for accurate registration,

[18] `http://www.prs.igp.ethz.ch/research/completed_projects/automatic_registration_of_point_clouds.html`

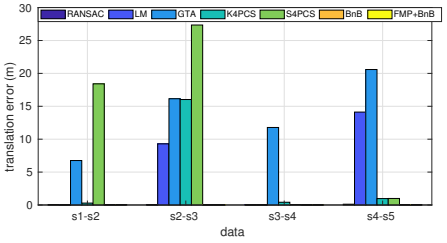
[19] Note, the threshold refers to errors after initial alignment from scratch. Of course, the result can be refined with ICP-type methods that are based on all points, not just a sparse match set.



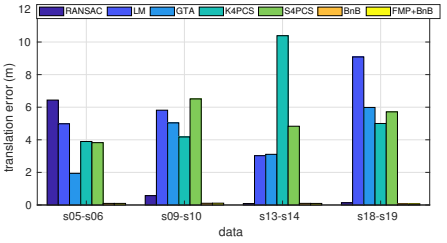
(a) Rotation error—*Arch.*



(b) Rotation error—*Facility.*

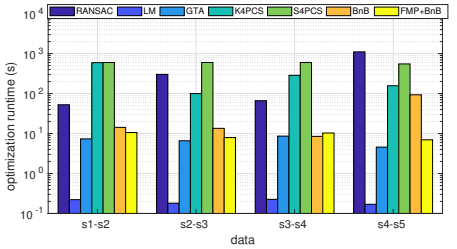


(c) Translation error—*Arch.*

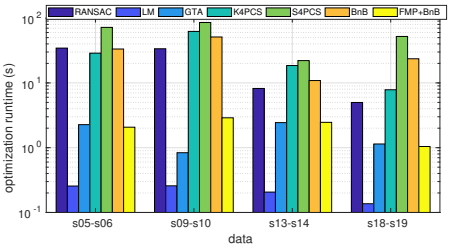


(d) Translation error—*Facility.*

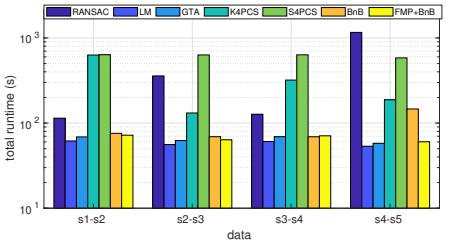
Figure 10: The accuracy of all registration methods on real-world data.



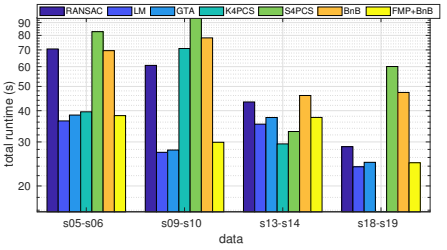
(a) Runtime for optimization—*Arch.*



(b) Runtime for optimization—*Facility.*

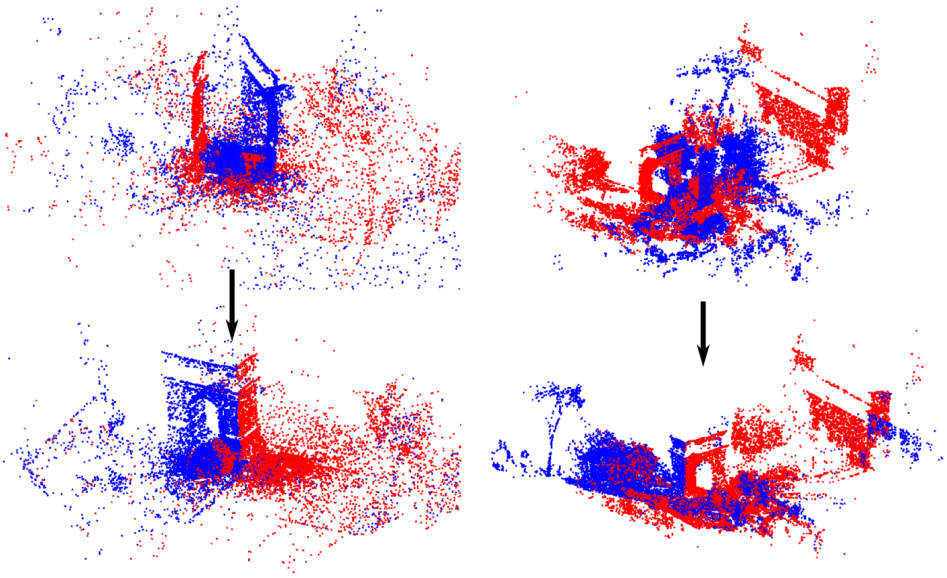


(c) Total runtime—*Arch.*



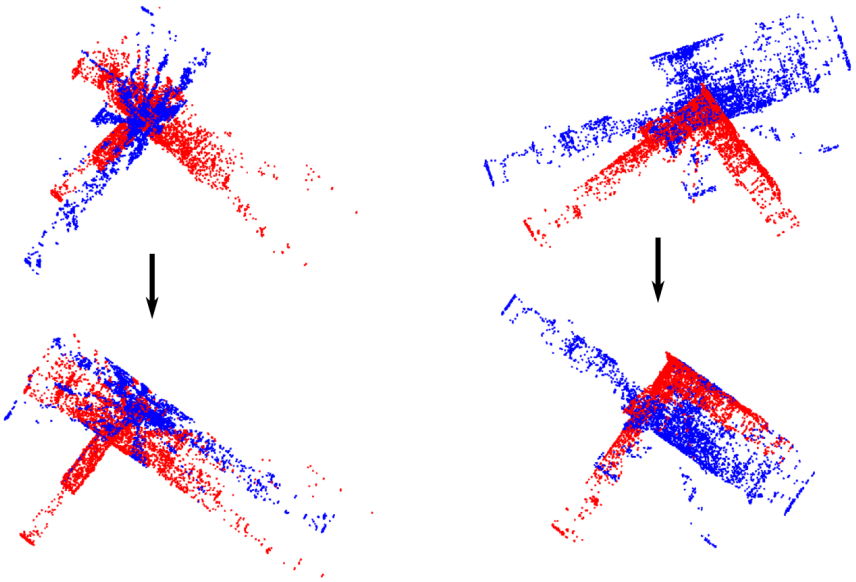
(d) Total runtime—*Facility.*

Figure 11: The log scaled runtime of all methods with (up) and without (bottom) input genreration (only generating keypoints for K4PCS and S4PCS) on real-world data.



(a) s3-s4—*Arch.*

(b) s4-s5—*Arch.*



(c) s9-s10—*Facility.*

(d) s18-s19—*Facility.*

Figure 12: Registered pairwise real-world data. 10k points are shown for each point cloud.

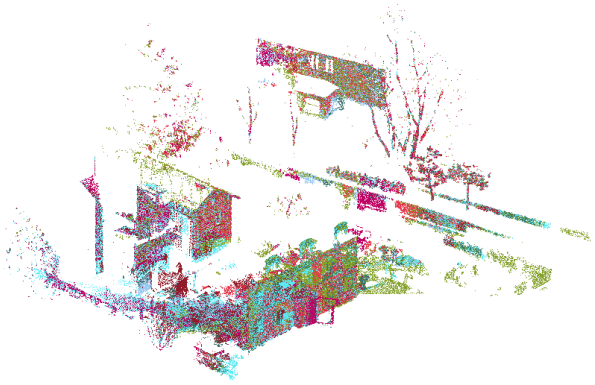


Figure 13: Complete registration result—*Facade*. All 7 scans were registered in 134.95s.

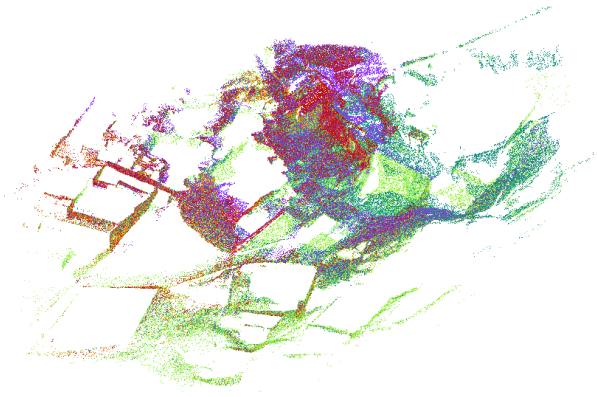


Figure 14: Complete registration result—*Courtyard*. All 8 scans were registered in 84.02s.

so long as they are correct. Given some set of candidate correspondences that may be contaminated with a large number of incorrect matches (which is often the case for real 3D point clouds); our algorithm first applies a fast pruning method that leaves the optimum of the alignment objective unchanged, then uses the reduced point set to find that optimum with the branch-and-bound method. The pruning greatly accelerates the solver while keeping intact the optimality w.r.t. the original problem. The BnB solver explicitly searches the 3D translation space, which can be bounded efficiently; while solving for 1D rotation implicitly with a novel, polynomial-time algorithm. Experiments show that our algorithm is significantly more reliable than previous methods, with competitive speed.

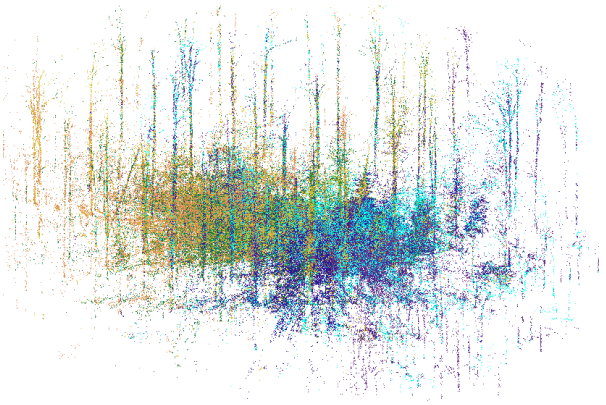


Figure 15: Complete registration result—*Trees*. All 6 scans were registered in 142.72s.

8. Acknowledgement

This work was supported by the ARC grant DP160103490.

9. Appendix

9.1. Compute $\texttt{int}_i$ and solve the max-stabbing problem

Algorithm 4 shows how to compute $\texttt{int}_i = [\alpha_i, \beta_i]$ for each $(\mathbf{p}_i, \tilde{\mathbf{q}}_i)$ during rotation search. As shown in Figure 16, $\texttt{circ}_i$ and $\texttt{ball}_i(\epsilon)$ intersect if and only if the closest distance d_i from $\tilde{\mathbf{q}}_i$ to $\texttt{circ}_i$ is within ϵ, where

$$d_i = \sqrt{(\|\mathbf{p}_i\|_{xy} - \|\tilde{\mathbf{q}}_i\|_{xy})^2 + (\mathbf{p}_i(3) - \tilde{\mathbf{q}}_i(3))^2}. \quad (28)$$

In the above equation, $\mathbf{p}_i(3)$ is the 3rd channel of $\mathbf{p}_i$ and $\|\mathbf{p}_i\|_{xy} = \sqrt{\mathbf{p}_i(1)^2 + \mathbf{p}_i(2)^2}$ is the *horizontal* length of $\mathbf{p}_i$. And the two intersecting points of $\texttt{circ}_i$ and $\texttt{ball}_i(\epsilon)$, namely $\mathbf{R}(\alpha_i)\mathbf{p}_i$ and $\mathbf{R}(\beta_i)\mathbf{p}_i$, have the same *azimuthal* angular distance γ to $\tilde{\mathbf{q}}_i$. And by

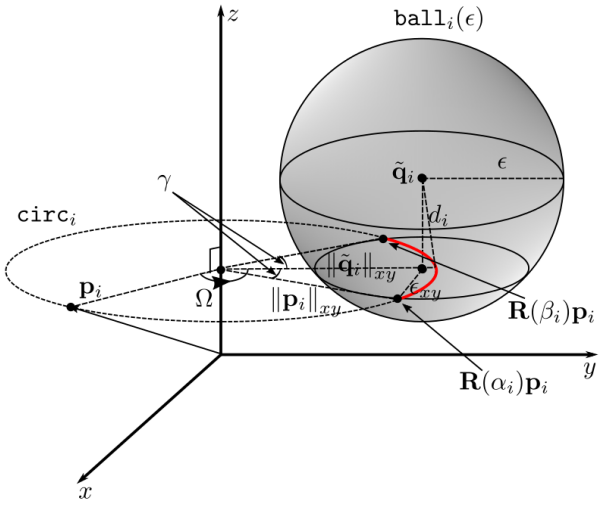


Figure 16: Illustration of Algorithm 4, $\texttt{int}_i$ is rendered in red.

Algorithm 4 Compute the angle interval that aligns $\mathbf{p}_i$ with $\tilde{\mathbf{q}}_i$

Require: $\mathbf{p}_i$, $\tilde{\mathbf{q}}_i$ and ϵ, such that $d_i \le \epsilon$.

1: $\epsilon_{xy} \leftarrow \sqrt{\epsilon^2 - (\mathbf{p}_i(3) - \tilde{\mathbf{q}}_i(3))^2}$
2: **if** $|\|\mathbf{p}_i\|_{xy} + \|\tilde{\mathbf{q}}_i\|_{xy}| \le \epsilon_{xy}$ **then**
3: **return** $[0, 2\pi]$.
4: **else**
5: $\Omega = azi(\tilde{\mathbf{q}}_i) - azi(\mathbf{p}_i), \gamma = \arccos \frac{\|\mathbf{p}_i\|_{xy}^2 + \|\tilde{\mathbf{q}}_i\|_{xy}^2 - \epsilon_{xy}}{2\|\mathbf{p}_i\|_{xy}\|\tilde{\mathbf{q}}_i\|_{xy}}$.
6: **return** $[\Omega - \gamma, \Omega + \gamma]$.
7: **end if**

computing the azimuth Ω from $\mathbf{p}_i$ to $\tilde{\mathbf{q}}_i$, i.e., $\Omega = azi(\tilde{\mathbf{q}}_i) - azi(\mathbf{p}_i)$, where $azi(\cdot)$ is the azimuth of a point, $\texttt{int}_i$ is simply $[\Omega - \gamma, \Omega + \gamma]$. Note that when $\texttt{circ}_i$ is inside $\texttt{ball}_i(\epsilon)$ (Line 2), $\texttt{int}_i$ is $[0, 2\pi]$.

γ is computed by the law of cosines[20], because it is an interior angle of the triangle whose three edge lengths are $\|\mathbf{p}_i\|_{xy}$, $\|\tilde{\mathbf{q}}_i\|_{xy}$ and ϵ_{xy}, where ϵ_{xy} is the *horizontal* distance between $\tilde{\mathbf{q}}_i$ and $\mathbf{R}(\alpha_i)\mathbf{p}_i$ (or $\mathbf{R}(\beta_i)\mathbf{p}_i$).

After computing all $\texttt{int}_i$'s, the max-stabbing problem (11) can be efficiently solved by Algorithm 5. Observe that one of the endpoints among all $\texttt{int}_i$'s must achieve the max-stabbing, the idea of Algorithm 5 is to compute the stabbing value δ for all endpoints and find the maximum one.

[20] https://en.wikipedia.org/wiki/Law_of_cosines

Algorithm 5 Max-Stabbing algorithm for 1D rotation estimation

Require: $U = \{\texttt{int}_i\}_{i=1}^M$, where $\texttt{int}_i = [\alpha_i, \beta_i]$.

1: $\mathbf{V} \leftarrow \bigcup_{i=1}^{M} \{[\alpha_i, 0], [\beta_i, 1]\}$, sort $\mathbf{V}$ ascendingly according to **a1** and **a2**.
2: $\tilde{\delta} \leftarrow 0$, $\delta \leftarrow 0$.
3: **for** each $\mathbf{v} \in \mathbf{V}$ **do**
4: **if** $\mathbf{v}(2) = 0$ **then**
5: $\delta \leftarrow \delta + 1$. And if $\delta > \tilde{\delta}$, then $\tilde{\delta} \leftarrow \delta, \tilde{\theta} \leftarrow \mathbf{v}(1)$.
6: **else**
7: $\delta \leftarrow \delta - 1$.
8: **end if**
9: **end for**
10: **return** $\tilde{\delta}, \tilde{\theta}$.

We first pack all endpoints into an array $\mathbf{V} = \bigcup_{i=1}^{M} \{[\alpha_i, 0], [\beta_i, 1]\}$. The 0/1 label attached to each endpoint represents whether it is the end of an $\texttt{int}_i$, i.e., β_i. Then, to efficiently compute each δ, we sort $\mathbf{V}$ so that the endpoints with

a1: smaller angles;

a2: and the same angle but are the start of an $\texttt{int}_i$ (the assigned label is 0);

are moved to the front. After sorting, we initialize δ to 0 and traverse $\mathbf{V}$ from the first element to the last. When we "hit" the beginning of an $\texttt{int}_i$, we increment δ by 1, and when we "hit" the end of an $\texttt{int}_i$, δ is reduced by 1. The max-stabbing value $\tilde{\delta}$ and its corresponding angle $\tilde{\theta}$ are returned in the end.

Since the time complexity for sorting and traversing $\mathbf{V}$ is respectively $\mathcal{O}(M \log M)$ and $\mathcal{O}(M)$, the one for Algorithm 5 is $\mathcal{O}(M \log M)$. The space complexity of $\mathcal{O}(M)$ can be achieved with advanced sorting

algorithms like merge sort[21].

9.2. Proof of Lemma 1

First, to prove (15), we re-express $\mathbf{t}^*(\mathbb{S}) = \arg\max_{\mathbf{t}\in\mathbb{S}} U(\mathbf{t} \mid \mathcal{C}, \epsilon)$ as $\mathbf{t}_\mathbb{S}+\mathbf{t}^{*\prime}(\mathbb{S})$, where $\|\mathbf{t}^{*\prime}(\mathbb{S})\| \leq d_\mathbb{S}$. Using this re-expression, we have

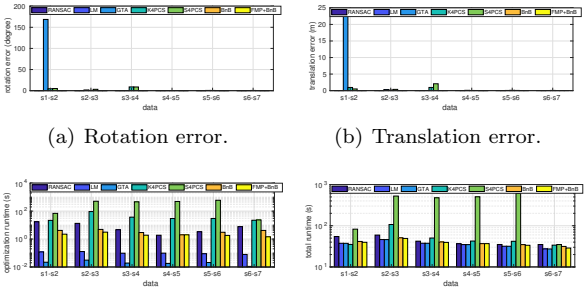
$$\|\mathbf{R}(\theta)\mathbf{p}_i + \mathbf{t}_\mathbb{S} + \mathbf{t}^{*\prime}(\mathbb{S}) - \mathbf{q}_i\| \leq \epsilon \tag{29a}$$
$$\Rightarrow \|\mathbf{R}(\theta)\mathbf{p}_i + \mathbf{t}_\mathbb{S} - \mathbf{q}_i\| - \|\mathbf{t}^{*\prime}(\mathbb{S})\| \leq \epsilon \tag{29b}$$
$$\Rightarrow \|\mathbf{R}(\theta)\mathbf{p}_i + \mathbf{t}_\mathbb{S} - \mathbf{q}_i\| - d_\mathbb{S} \leq \epsilon \tag{29c}$$
$$\Leftrightarrow \|\mathbf{R}(\theta)\mathbf{p}_i + \mathbf{t}_\mathbb{S} - \mathbf{q}_i\| \leq \epsilon + d_\mathbb{S}. \tag{29d}$$

(29a) to (29b) is due to the triangle inequality. According to (29a) and (29d), as long as $(\mathbf{p}_i, \mathbf{q}_i)$ contributes 1 in $\max_{\mathbf{t}\in\mathbb{S}} U(\mathbf{t} \mid \mathcal{C}, \epsilon) = U(\mathbf{t}_\mathbb{S} + \mathbf{t}^{*\prime}(\mathbb{S}) \mid \mathcal{C}, \epsilon)$, it must also contribute 1 in $U(\mathbf{t}_\mathbb{S} \mid \mathcal{C}, \epsilon + d_\mathbb{S})$, i.e., $U(\mathbf{t}_\mathbb{S} \mid \mathcal{C}, \epsilon + d_\mathbb{S}) \geq \max_{\mathbf{t}\in\mathbb{S}} U(\mathbf{t} \mid \mathcal{C}, \epsilon)$.

And when $\mathbb{S}$ tends to a point $\mathbf{t}$, $d_\mathbb{S}$ tends to 0. Therefore, $\bar{U}(\mathbb{S} \mid \mathcal{C}, \epsilon) = U(\mathbf{t}_\mathbb{S} \mid \mathcal{C}, \epsilon + d_\mathbb{S}) \rightarrow U(\mathbf{t} \mid \mathcal{C}, \epsilon)$.

9.3. Further results

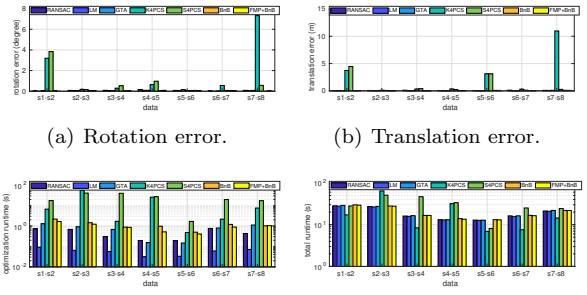
Figure 17 to 19 show the accuracy and runtime of our method and all other competitors in Section 6.2 for *Facade*, *Courtyard* and *Trees*. The inlier threshold ϵ were set to 0.05 m for *Facade*, 0.2 m for *Courtyard* and 0.2 m for *Trees*.



(a) Rotation error. (b) Translation error.

(c) Runtime for optimization. (d) Total runtime.

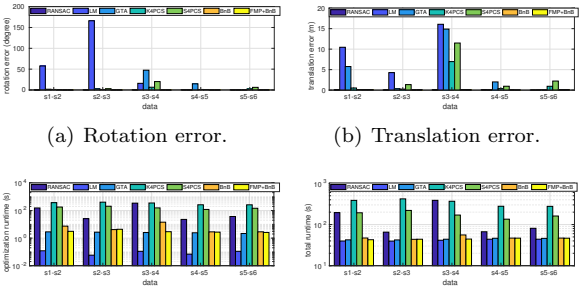
Figure 17: Accuracy and log scaled runtime for *Facade*.



(a) Rotation error. (b) Translation error.

(c) Runtime for optimization. (d) Total runtime.

Figure 18: Accuracy and log scaled runtime for *Courtyard*.



(a) Rotation error. (b) Translation error.

(c) Runtime for optimization. (d) Total runtime.

Figure 19: Accuracy and log scaled runtime for *Trees*.

[21] https://en.wikipedia.org/wiki/Merge_sort

References

Aiger, D., Mitra, N. J., Cohen-Or, D., 2008. 4-points congruent sets for robust pairwise surface registration. In: ACM Transactions on Graphics (TOG). Vol. 27. ACM, p. 85.

Akca, D., 2003. Full automatic registration of laser scanner point clouds. Tech. rep., ETH Zurich.

Albarelli, A., Rodola, E., Torsello, A., 2010. A game-theoretic approach to fine surface registration without initial motion estimation. In: Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on. IEEE, pp. 430–437.

Bazin, J.-C., Seo, Y., Pollefeys, M., 2012. Globally optimal consensus set maximization through rotation search. In: Asian Conference on Computer Vision. Springer, pp. 539–551.

Besl, P. J., McKay, N. D., 1992. Method for registration of 3-d shapes. In: Sensor Fusion IV: Control Paradigms and Data Structures. Vol. 1611. International Society for Optics and Photonics, pp. 586–607.

Brenner, C., Dold, C., 2007. Automatic relative orientation of terrestrial laser scans using planar structures and angle constraints. In: ISPRS Workshop on Laser Scanning. Vol. 200. pp. 84–89.

Breuel, T. M., 2001. A practical, globally optimal algorithm for geometric matching under uncertainty. Electronic Notes in Theoretical Computer Science 46, 188–202.

Bustos, A. P., Chin, T.-J., 2017. Guaranteed outlier removal for point cloud registration with correspondences. IEEE Transactions on Pattern Analysis and Machine Intelligence.

Campbell, D., Petersson, L., 2016. Gogma: Globally-optimal gaussian mixture alignment. In: CVPR. pp. 5685–5694.

Chen, C.-S., Hung, Y.-P., Cheng, J.-B., 1999. Ransac-based darces: A new approach to fast automatic registration of partially overlapping range images. IEEE Transactions on Pattern Analysis and Machine Intelligence 21 (11), 1229–1234.

Chin, T.-J., Cai, Z., Neumann, F., 2018. Robust fitting in computer vision: Easy or hard? arXiv preprint arXiv:1802.06464.

Chin, T.-J., Suter, D., 2017. The maximum consensus problem: Recent algorithmic advances. Synthesis Lectures on Computer Vision 7 (2), 1–194.

De Berg, M., Van Kreveld, M., Overmars, M., Schwarzkopf, O. C., 2000. Computational geometry. In: Computational geometry. Springer, pp. 1–17.

Drost, B., Ulrich, M., Navab, N., Ilic, S., 2010. Model globally, match locally: Efficient and robust 3d object recognition. In: Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on. Ieee, pp. 998–1005.

Fischler, M. A., Bolles, R. C., 1981. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Communications of the ACM 24 (6), 381–395.

Franaszek, M., Cheok, G. S., Witzgall, C., 2009. Fast automatic registration of range images from 3d imaging systems using sphere targets. Automation in Construction 18 (3), 265–274.

Głomb, P., 2009. Detection of interest points on 3d data: Extending the harris operator. In: Computer Recognition Systems 3. Springer, pp. 103–111.

Harris, C., Stephens, M., 1988. A combined corner and edge detector. In: Alvey vision conference. Vol. 15. Citeseer, pp. 10–5244.

Houshiar, H., Elseberg, J., Borrmann, D., Nüchter, A., 2015. A study of projections for key point based registration of panoramic terrestrial 3d laser scan. Geo-spatial Information Science 18 (1), 11–31.

Lee, K., Woo, H., Suk, T., 2001. Data reduction methods for reverse engineering. The International Journal of Advanced Manufacturing Technology 17 (10), 735–743.

Lowe, D. G., 1999. Object recognition from local scale-invariant features. In: Computer vision, 1999. The proceedings of the seventh IEEE international conference on. Vol. 2. Ieee, pp. 1150–1157.

Lu, F., Milios, E., 1997. Globally consistent range scan alignment for environment mapping. Autonomous robots 4 (4), 333–349.

Mellado, N., Aiger, D., Mitra, N. J., 2014. Super 4pcs fast global pointcloud registration via smart indexing. In: Computer Graphics Forum. Vol. 33. Wiley Online Library, pp. 205–215.

Parra Bustos, A., Chin, T.-J., 2015. Guaranteed outlier removal for rotation search. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 2165–2173.

Parra Bustos, A., Chin, T.-J., Eriksson, A., Li, H., Suter, D., 2016. Fast rotation search with stereographic projections for 3d registration. IEEE Transactions on Pattern Analysis and Machine Intelligence 38 (11), 2227–2240.

Pomerleau, F., Colas, F., Siegwart, R., Magnenat, S., 2013. Comparing icp variants on real-world data sets. Autonomous Robots 34 (3), 133–148.

Rabbani, T., Dijkman, S., van den Heuvel, F., Vosselman, G., 2007. An integrated approach for modelling and global registration of point clouds. ISPRS journal of Photogrammetry and Remote Sensing 61 (6), 355–370.

Rusinkiewicz, S., Levoy, M., 2001. Efficient variants of the icp algorithm. In: 3-D Digital Imaging and Modeling, 2001. Proceedings. Third International Conference on. IEEE, pp. 145–152.

Rusu, R. B., Blodow, N., Beetz, M., 2009. Fast point feature histograms (fpfh) for 3d registration. In: Robotics and Automation, 2009. ICRA'09. IEEE International Conference on. IEEE, pp. 3212–3217.

Rusu, R. B., Blodow, N., Marton, Z. C., Beetz, M., 2008. Aligning point cloud views using persistent feature histograms. In: Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on. IEEE, pp. 3384–3391.

Scovanner, P., Ali, S., Shah, M., 2007. A 3-dimensional sift descriptor and its application to action recognition. In: Proceedings of the 15th ACM international conference on Multimedia. ACM, pp. 357–360.

Svarm, L., Enqvist, O., Oskarsson, M., Kahl, F., 2014. Accurate localization and pose estimation for large 3d models. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 532–539.

Theiler, P. W., Wegner, J. D., Schindler, K., 2014. Keypoint-based 4-points congruent sets–automated marker-less registration of laser scans. ISPRS journal of photogrammetry and remote sensing 96, 149–163.

Theiler, P. W., Wegner, J. D., Schindler, K., 2015. Globally consistent registration of terrestrial laser scans via graph optimization. ISPRS Journal of Photogrammetry and Remote Sensing 109, 126–138.

Turk, G., Levoy, M., 1994. Zippered polygon meshes from range images. In: Proceedings of the 21st annual conference on Computer graphics and interactive techniques. ACM, pp. 311–318.

Yang, J., Li, H., Campbell, D., Jia, Y., 2016. Go-icp: a globally optimal solution to 3d icp point-set registration. IEEE transactions on pattern analysis and machine intelligence 38 (11), 2241–2254.

Yu, C., Ju, D. Y., 2018. A maximum feasible subsystem for globally optimal 3d point cloud registration. Sensors 18 (2), 544.

Zhong, Y., 2009. Intrinsic shape signatures: A shape descriptor for 3d object recognition. In: Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on. IEEE, pp. 689–696.

Zhou, Q.-Y., Park, J., Koltun, V., 2016. Fast global registration. In: European Conference on Computer Vision. Springer, pp. 766–782.

Chapter 7

Conclusion

In previous chapters, we proposed several new theories of, and algorithms for, consensus maximization. In particular, the NP-hardness result in Chapter 3 tells us that a general, tractable and globally optimal solver for consensus maximization does not exist, given that it is likely that P $\neq$ NP. The APX-hardness prohibits even approximate solvers. Note that none of the existing efficient algorithms, including RANSAC variants and deterministic optimization approaches, can guarantee an approximation ratio, which is required for them to become an approximate solver. Though slightly negative, these theories motivates several interesting research directions:

1. Replace/complement RANSAC variants with deterministic algorithms that conduct efficient local optimization. Chapter 4 presented an effective approach of this type.

2. Use structures that exist in special applications to simplify the problem, such that we can develop an application-oriented, efficient and optimal algorithm. For example, in Chapter 6, we utilized the rotation axis obtained by the sensors to reduce the rotation search problem from 3D to 1D. And we demonstrated that 1D consensus maximization (for rotation search) can be efficiently solved by a polynomial-time algorithm.

3. Design pre-processing algorithms (such as [8, 31] and FMP in Chapter 6) that can efficiently identify (some but not necessarily all) *true outliers*, such that we can reduce the problem size for consensus maximization without affecting the optimal solution.

Chapter 3 also proved the prametrized-complexity, i.e., consensus maximization is W[1]-hard in the problem dimension, but FPT in the number of outliers. W[1]-hard means that the problem is not FPT. And being FPT in a certain parameter means that as

long as the value of this parameter is small, consensus maximization can still be solved efficiently. Therefore, choosing the right parametrization is important. Currently, the only available parameter for FPT algorithms is *the number of outliers*. More efficient algorithms may naturally appear if better parametrizations can be discovered. Meanwhile, the development of better FPT algorithms for the existing parametrization, i.e., number of outliers, is also important. The work of Chapter 5 significantly accelerates the tree search algorithm, which represents a solid step in that direction.

7.1 Limitation and future work

The algorithms proposed in this thesis are mainly designed for single model fitting. An interesting future research direction is to consider the case of multi-model fitting. In the meantime, experiments done in the previous chapters are mostly on small/medium scale problems, such as two-view geometry and pair-wise point cloud registration. Extending the proposed algorithms to larger scales (e.g., for large scale structure-from-motion with a large number of views) is also an important future direction.

Bibliography

[1] https://en.wikipedia.org/wiki/Outlier#cite_note\protect\discretionary{\char\hyphenchar\font}{}{}1.

[2] https://en.wikipedia.org/wiki/Least_squares.

[3] https://en.wikipedia.org/wiki/Maximum_likelihood_estimation.

[4] https://en.wikipedia.org/wiki/M-estimator.

[5] https://en.wikipedia.org/wiki/Redescending_M-estimator.

[6] https://en.wikipedia.org/wiki/Parameterized_complexity.

[7] K. Aftab and R. Hartley. "Convergence of iteratively re-weighted least squares to robust m-estimators". In: *2015 IEEE Winter Conference on Applications of Computer Vision*. IEEE. 2015, pp. 480–487.

[8] T.-J. Chin, Y. H. Kee, A. Eriksson, and F. Neumann. "Guaranteed outlier removal with mixed integer linear programs". In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[9] T.-J. Chin, P. Purkait, A. Eriksson, and D. Suter. "Efficient globally optimal consensus maximisation with tree search". In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015.

[10] T.-J. Chin, P. Purkait, A. Eriksson, and D. Suter. "Efficient globally optimal consensus maximisation with tree search". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 39.4 (2017), pp. 758–772.

[11] O. Chum and J. Matas. "Matching with PROSAC-progressive sample consensus". In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. Vol. 1. IEEE. 2005, pp. 220–226.

[12] O. Chum, J. Matas, and J. Kittler. "Locally optimized RANSAC". In: *Joint Pattern Recognition Symposium*. Springer. 2003, pp. 236–243.

[13] J. Clausen. "Branch and bound algorithms-principles and examples". In: *Department of Computer Science, University of Copenhagen* (1999), pp. 1–30.

[14] N. Dalal and B. Triggs. "Histograms of oriented gradients for human detection". In: *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*. Vol. 1. IEEE. 2005, pp. 886–893.

[15] O. Enqvist, E. Ask, F. Kahl, and K. Åström. "Robust fitting for multiple view geometry". In: *European Conference on Computer Vision (ECCV)*. 2012.

[16] M. A. Fischler and R. C. Bolles. "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography". In: *Communications of the ACM* 24.6 (1981), pp. 381–395.

[17] F. R. Hampel, E. M. Ronchetti, P. J. Rousseeuw, and W. A. Stahel. *Robust statistics: the approach based on influence functions*. Vol. 196. John Wiley & Sons, 2011.

[18] P. J. Huber. "Robust estimation of a location parameter". In: *Breakthroughs in statistics*. Springer, 1992, pp. 492–518.

[19] A. Land and A. Doig. "An Automatic Method of Solving Discrete Programming Problems". In: *Econometrica: Journal of the Econometric Society* (1960), pp. 497–520.

[20] H. Le, T.-J. Chin, and D. Suter. "An exact penalty method for locally convergent maximum consensus". In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[21] H. Le, A. P. Eriksson, T.-T. Do, T.-J. Chin, and D. Suter. "Non-smooth M-estimator for Maximum Consensus Estimation." In: *BMVC*. 2018, p. 4.

[22] H. M. Le, T.-J. Chin, A. Eriksson, T.-T. Do, and D. Suter. "Deterministic approximate methods for maximum consensus robust fitting". In: *IEEE transactions on pattern analysis and machine intelligence* (2019).

[23] K. Lebeda, J. Matas, and O. Chum. "Fixing the locally optimized ransac–full experimental evaluation". In: *British machine vision conference*. Citeseer. 2012, pp. 1–11.

[24] H. Li. "Consensus set maximization with guaranteed global optimality for robust geometry estimation". In: *IEEE International Conference on Computer Vision (ICCV)*. 2009.

[25] H. Li. "A practical algorithm for L triangulation with outliers". In: *2007 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2007, pp. 1–8.

[26] H. Li. "Consensus set maximization with guaranteed global optimality for robust geometry estimation". In: *2009 IEEE 12th International Conference on Computer Vision*. IEEE. 2009, pp. 1074–1080.

[27] D. G. Lowe. "Object recognition from local scale-invariant features". In: *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*. Vol. 2. Ieee. 1999, pp. 1150–1157.

[28] J. Matoušek. "On geometric optimization with few violated constraints". In: *Discrete and Computational Geometry* 14.4 (1995), pp. 365–384.

[29] C. Olsson, O. Enqvist, and F. Kahl. "A polynomial-time bound for matching and registration with outliers". In: *2008 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2008, pp. 1–8.

[30] C. Olsson, A. P. Eriksson, and F. Kahl. "Efficient optimization for L_∞-problems using pseudoconvexity". In: *2007 IEEE 11th International Conference on Computer Vision*. IEEE. 2007, pp. 1–8.

[31] A. Parra Bustos and T.-J. Chin. "Guaranteed outlier removal for rotation search". In: *IEEE International Conference on Computer Vision (ICCV)*. 2015.

[32] P. Purkait, C. Zach, and A. Eriksson. "Maximum Consensus Parameter Estimation by Reweighted ℓ_1 Methods". In: *International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition*. Springer. 2017, pp. 312–327.

[33] P. J. Rousseeuw. "Least median of squares regression". In: *Journal of the American statistical association* 79.388 (1984), pp. 871–880.

[34] P. Rousseeuw and A. Leroy. "Robust regression and outlier detection". In: (1987).

[35] R. B. Rusu, N. Blodow, and M. Beetz. "Fast point feature histograms (FPFH) for 3D registration". In: *2009 IEEE international conference on robotics and automation*. IEEE. 2009, pp. 3212–3217.

[36] R. B. Rusu, N. Blodow, Z. C. Marton, and M. Beetz. "Aligning point cloud views using persistent feature histograms". In: *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2008, pp. 3384–3391.

[37] M. Sharir and E. Welzl. "A combinatorial bound for linear programming and related problems". In: *Annual Symposium on Theoretical Aspects of Computer Science*. Springer. 1992, pp. 567–579.

[38] B. J. Tordoff and D. W. Murray. "Guided-MLESAC: Faster image transform estimation by using matching priors". In: *IEEE transactions on pattern analysis and machine intelligence* 27.10 (2005), pp. 1523–1535.

[39] Q.-H. Tran, T.-J. Chin, W. Chojnacki, and D. Suter. "Sampling minimal subsets with large spans for robust estimation". In: *International Journal of Computer Vision (IJCV)* 106.1 (2014), pp. 93–112.

[40] R. W. Wedderburn. "Quasi-likelihood functions, generalized linear models, and the Gauss—Newton method". In: *Biometrika* 61.3 (1974), pp. 439–447.

[41] Y. Zheng, S. Sugimoto, and M. Okutomi. "Deterministically maximizing feasible subsystems for robust model fitting with unit norm constraints". In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*. 2011.